

# O Produto e o Processo

**Engenharia de Software**  
**Profa. Inês A.G.Boaventura**  
**2. Semestre/2005**

# O Produto

## **Dois papéis do Software:**

- produto**

(quando libera o potencial embutido no hardware)

- veículo para liberar produto**

(quando ele atua como uma base para o controle do computador - sistema operacional; para a comunicação de informação - redes; para a criação e controle de outros programas)

# Evolução do Software

## (1950 - 1965)

- ⇒ O hardware sofreu contínuas mudanças
- ⇒ O software era uma arte "secundária" para a qual havia poucos métodos sistemáticos
- ⇒ O hardware era de propósito geral
- ⇒ O software era específico para cada aplicação
- ⇒ Não havia documentação

# Evolução do Software

## (1965 - 1975)

- ⇒ Multiprogramação e sistemas multiusuários
- ⇒ Técnicas interativas
- ⇒ Sistemas de tempo real
- ⇒ 1ª geração de SGBD's
- ⇒ Produto de software - *software houses*
- ⇒ Bibliotecas de Software
- ⇒ Cresce nº de sistemas baseado em computador
- ⇒ Manutenção quase impossível

..... **CRISE DE SOFTWARE**

# Evolução do Software

**(1975 - *hoje*)**

- ⇒ Sistemas distribuídos
- ⇒ Redes locais e globais
- ⇒ Uso generalizado de microprocessadores - produtos inteligentes
- ⇒ Hardware de baixo custo
- ⇒ Impacto de consumo

# Evolução do Software

## (Quarta era do software: atualidade)

- Tecnologias orientadas o objetos
- Sistemas especialistas e software de inteligência artificial usados na prática
- Software de rede neural artificial
- Computação Paralela
- Internet

# Problemas que Persistem e se Intensificam

## Apesar da evolução do software...

- A habilidade em construir software deixa a desejar em relação ao potencial do hardware
- A construção de software não é rápida o suficiente para atender as necessidades do mercado
- A sociedade depende cada vez mais de software confiável; quando ele falha, podem ocorrer gastos enormes e desgaste de muitos profissionais para arrumá-lo
- O esforço para construir software confiável e de qualidade é muito grande
- O suporte aos programas existentes é apoiado por projetos pobres e recursos inadequados

# Uma Perspectiva Industrial

**Hoje, é o software que custa mais do que o hardware.**

**Já há algum tempo, gerentes e técnicos se perguntam:**

- Porque é preciso tanto tempo para terminar os programas?
- Porque os custos são tão altos?
- Porque não se consegue encontrar todos os erros antes que o software seja liberado para os clientes?
- Porque existe uma dificuldade em medir o progresso à medida que o software está sendo construído?

**A preocupação em resolver essas questões tem levado à adoção das práticas da Engenharia de Software**



# Uma Fábrica de Software Envelhecida

- ↪ os sistemas de informação escritos há 20 anos, depois de incontáveis alterações estão hoje de uma forma que não permite manutenção (peq. mudanças  $\Rightarrow$  falha do sistema)
- ↪ aplicações de engenharia que geram dados de projeto críticos, devido à idade e reparos, não permitem que alguém entenda suas estruturas internas
- ↪ sistemas embutidos que possuem comportamentos inesperados, mas não podem se tornar inoperantes pois não há nada para substituí-los

# Competitividade do Software

- hoje o software é um negócio competitivo
- os principais direcionadores que propiciarão uma intensa competição na área de software são: custo, adequação de prazo e qualidade
- intensifica-se, portanto, uma rápida movimentação dos desenvolvedores para adotar práticas modernas de Engenharia de Software

# Software

## **1- Instruções**

quando executadas produzem a função e o desempenho desejados

## **2 - Estruturas de Dados**

possibilitam que os programas manipulem adequadamente a informação

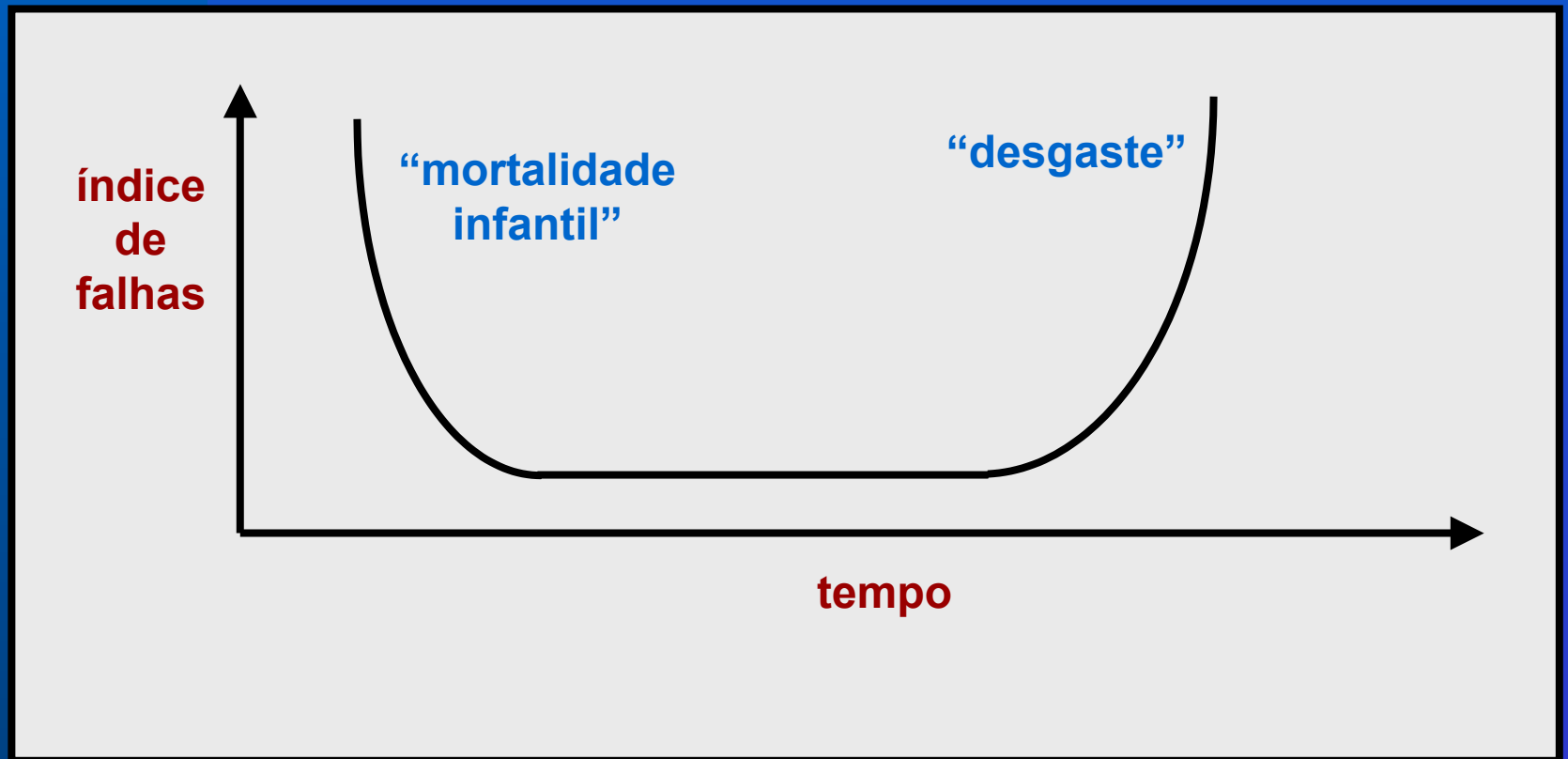
## **3 - Documentos**

descrevem a operação e o uso dos programas

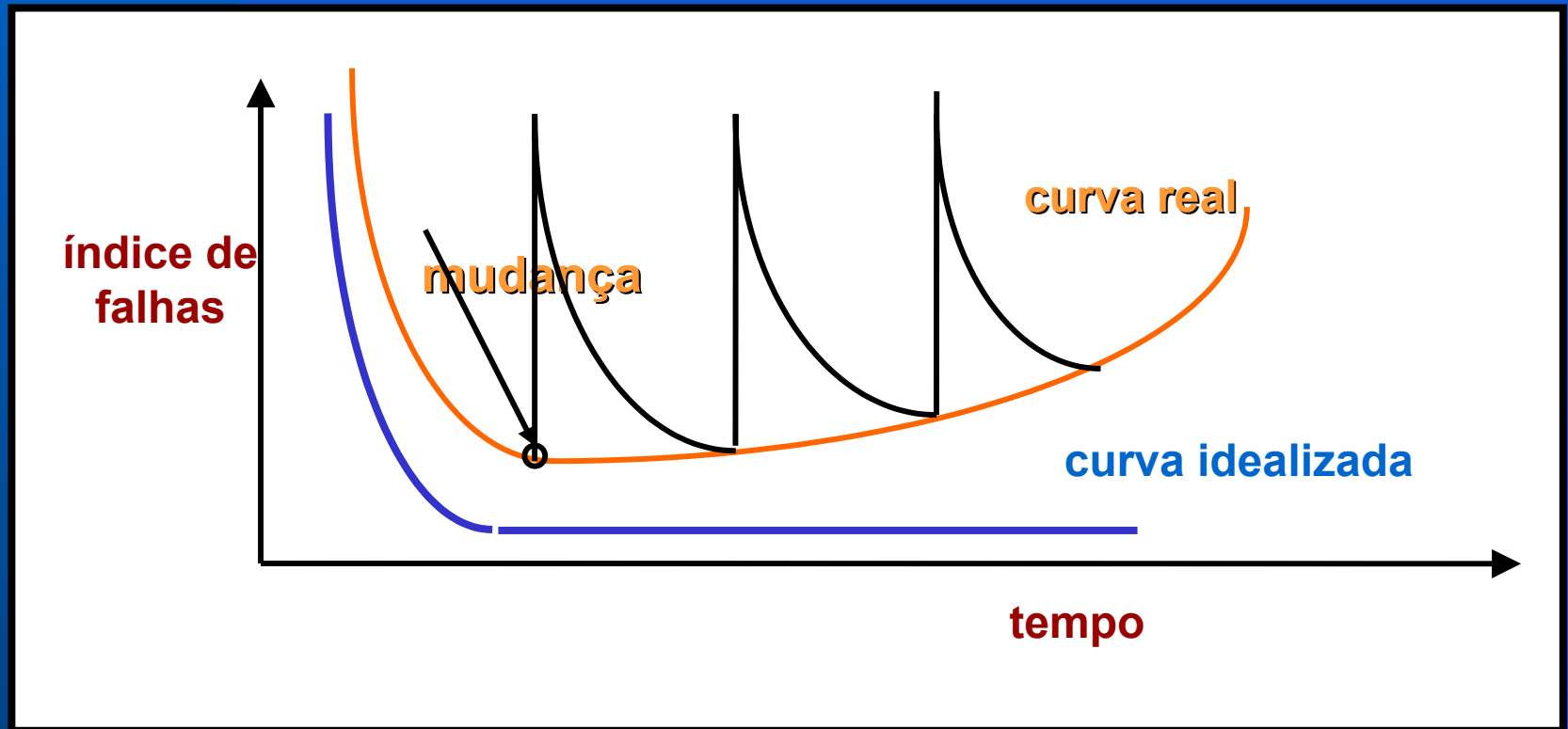
# Características do Software

1. desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico
2. não se desgasta mas se deteriora
3. a maioria é feita sob medida em vez de ser montada a partir de componentes existentes

# Curva de falhas para o Hardware



# Curva de falhas do Software



# Aplicações do Software

<b>BÁSICO</b>	programas de apoio a outros programas
<b>DE TEMPO REAL</b>	monitora, analisa e controla eventos do mundo real
<b>COMERCIAL</b>	operações comerciais e tomadas de decisões administrativas
<b>CIENTÍFICO E DE ENGENHARIA</b>	algoritmos de processamento de números
<b>EMBUTIDO</b>	controla produtos e sistemas de mercados industriais e de consumo
<b>DE COMPUTADOR PESSOAL</b>	processamento de textos, planilhas eletrônicas, diversões, etc.
<b>DE INTELIGÊNCIA ARTIFICIAL</b>	algoritmos não numéricos para resolver problemas que não sejam favoráveis à computação ou à análise direta

# Crise de Software

Refere-se a um conjunto de problemas encontrados no desenvolvimento de software:

**(1) As estimativas de prazo e de custo frequentemente são imprecisas**

“Não dedicamos tempo para coletar dados sobre o processo de desenvolvimento de software”

“Sem nenhuma indicação sólida de produtividade, não podemos avaliar com precisão a eficácia de novas ferramentas, métodos ou padrões”



# Crise de Software

**(2) A produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços**

“Os projetos de desenvolvimento de software normalmente são efetuados apenas com um vago indício das exigências do cliente”

# Crise de Software

## **(3) A qualidade de software às vezes é menos que adequada**

Só recentemente começam a surgir conceitos quantitativos sólidos de garantia de qualidade de software

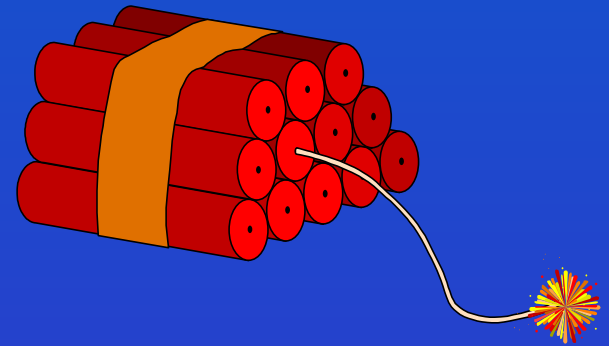
## **(4) O software existente é muito difícil de manter**

A tarefa de manutenção devora o orçamento destinado ao software

A facilidade de manutenção não foi enfatizada como um critério importante

# Crise de Software

- ✓ estimativas de prazo e de custo ↑
- ✓ produtividade das pessoas ↓
- ✓ qualidade de software ↓
- ✓ software difícil de manter ↑



# Causas dos problemas associados à Crise de Software

## 1. próprio caráter do Software

O software é um elemento de sistema lógico e não físico.

Conseqüentemente, o sucesso é medido pela qualidade de uma única entidade e não pela qualidade de muitas entidades manufaturadas

**O software não se desgasta, mas se deteriora!!!**

# Causas dos problemas associados à Crise de Software

## 2. falhas das pessoas responsáveis pelo desenvolvimento de Software

Gerentes sem nenhum *background* em software

Os profissionais da área de software têm recebido pouco treinamento formal em novas técnicas para o desenvolvimento de software

Resistência a mudanças

# Causas dos problemas associados à Crise de Software

## 3. mitos do Software



propagaram desinformação e confusão

- ▶ *administrativos*
- ▶ *cliente*
- ▶ *profissional*

# Mitos do Software (administrativos)

Já temos um manual repleto de padrões e procedimentos para a construção de software. Isso não oferecerá ao meu pessoal tudo o que eles precisam saber?

## **Realidade:**

**Será que o manual é usado?**

**Os profissionais sabem que ele existe?**

**Ele reflete a prática moderna de desenvolvimento de software?**

**Ele é completo?**

# Mitos do Software (administrativos)

Meu pessoal tem ferramentas de desenvolvimento de software de última geração; afinal lhes compramos os mais novos computadores.

**Realidade:**

É preciso muito mais do que os mais recentes computadores para se fazer um desenvolvimento de software de alta qualidade.



# Mitos do Software (administrativos)

Se nós estamos atrasados nos prazos, podemos adicionar mais programadores e tirar o atraso.

## **Realidade:**

O desenvolvimento de software não é um processo mecânico igual à manufatura.

Acrescentar pessoas em um projeto torna-o ainda mais atrasado. Pessoas podem ser acrescentadas, mas somente de uma forma planejada.

# Mitos do Software (cliente)

Uma declaração geral dos objetivos é suficiente para se começar a escrever programas - podemos preencher os detalhes mais tarde.

## **Realidade:**

**Uma definição inicial ruim é a principal causa de fracassos dos esforços de desenvolvimento de software.**

**É fundamental uma descrição formal e detalhada do domínio da informação, função, desempenho, interfaces, restrições de projeto e critérios de validação.**

# Mitos do Software (cliente)

Os requisitos de projeto modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, porque o software é flexível.

## **Realidade:**

Uma mudança, quando solicitada tardiamente num projeto, pode ser maior do que mais do que uma ordem de magnitude mais dispendiosa do que a mesma mudança solicitada nas fases iniciais.

# magnitude das mudanças

FASES	CUSTO DE MANUTENÇÃO
DEFINIÇÃO	1 x
DESENVOLVIMENTO	1.5 - 6x
MANUTENÇÃO	60 - 100x

# Mitos do Software (profissional)

Assim que escrevermos o programa e o colocarmos em funcionamento nosso trabalho estará completo.

## **Realidade:**

Os dados da indústria indicam que entre 50 e 70% de todo esforço gasto num programa serão despendidos depois que ele for entregue pela primeira vez ao cliente.

# Mitos do Software (profissional)

Enquanto não tiver o programa "funcionando", eu não terei realmente nenhuma maneira de avaliar sua qualidade.

## **Realidade:**

**Um dos mecanismos mais eficazes de garantia de qualidade de software pode ser aplicado a partir do início de um projeto - a *revisão técnica formal*.**

# O Processo

## O que é Processo de Software?

é um *conjunto de tarefas* requeridas para construir software de alta qualidade

“**Processo**” = Engenharia de Software ???

- + Processo define a abordagem e
- ou Engenharia de Software engloba também
- as tecnologias, como métodos e ferramentas

# Engenharia de Software

## Uma Tecnologia em Camadas

### Definição:

- (1) aplicação de uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção de software, ou seja, a aplicação da engenharia ao software
- (2) o estudo de abordagens do tipo declarado em (1)

[IEEE]



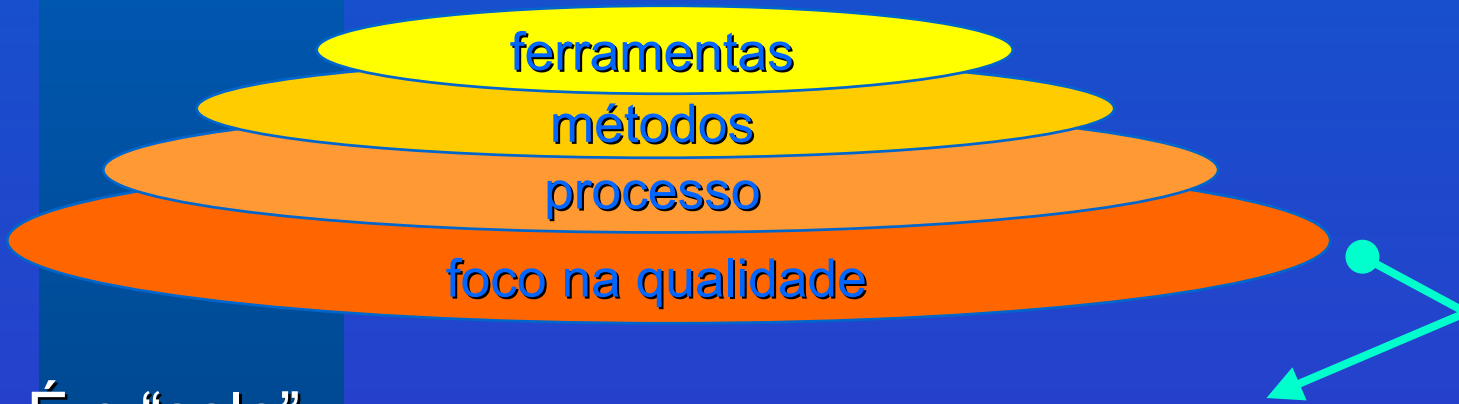
# Engenharia de Software

## Uma Tecnologia em Camadas



# Engenharia de Software

## Uma Tecnologia em Camadas

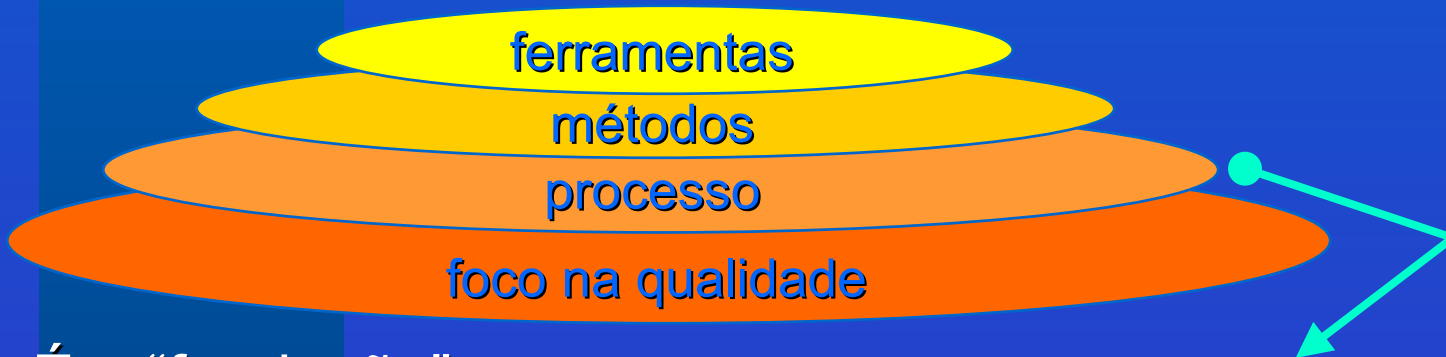


É o “solo”

Gerenciamento da Qualidade Total e filosofias similares produzem uma mudança cultural que permite o desenvolvimento crescente de abordagens mais maduras para a Engenharia de Software

# Engenharia de Software

## Uma Tecnologia em Camadas

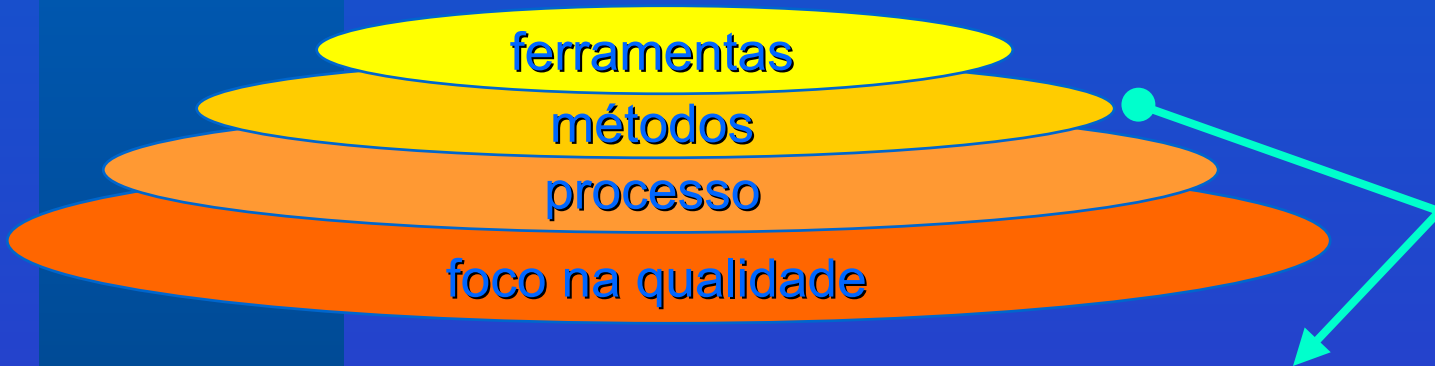


É a “fundação”

É a cola que gruda as camadas de tecnologias e permite um desenvolvimento de software racional e em tempo; define um conjunto de áreas chave do processo (KPA) que deve ser estabelecido para um uso efetivo da Engenharia de Software

# Engenharia de Software

## Uma Tecnologia em Camadas

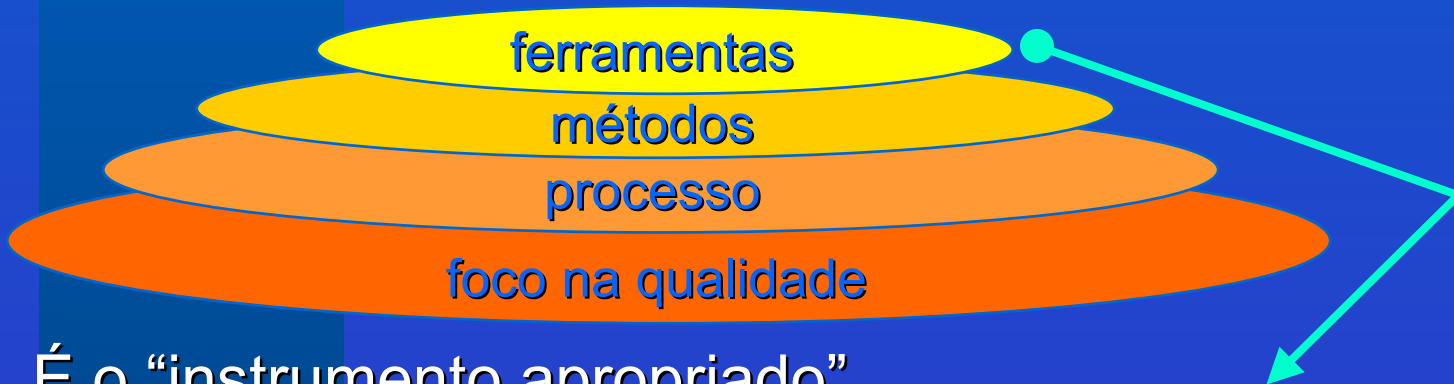


É o “como fazer”

Englobam um conjunto de tarefas que inclui análise de requisitos, projeto, construção de programas, teste e manutenção

# Engenharia de Software

## Uma Tecnologia em Camadas



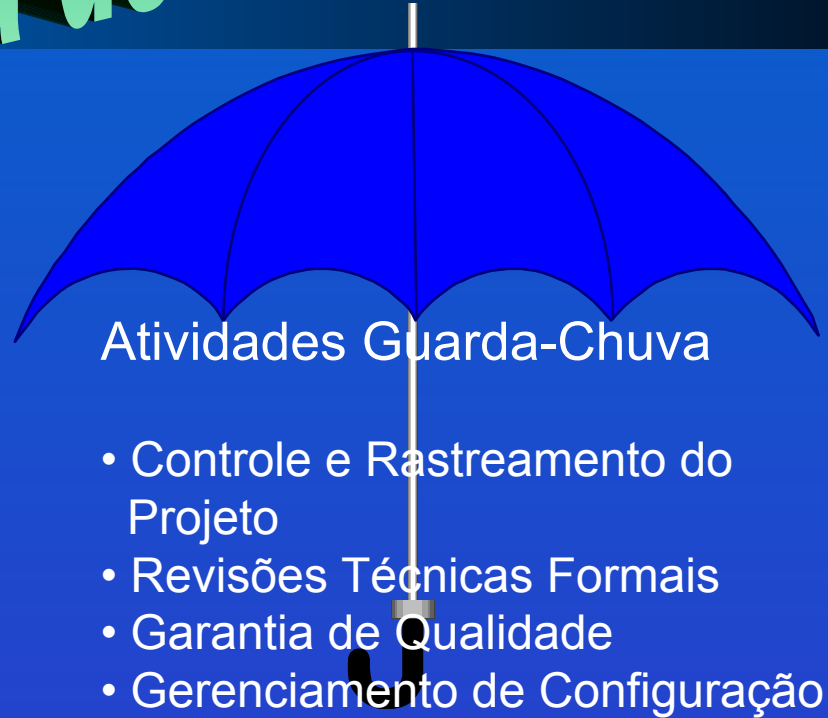
É o “instrumento apropriado”

Dão suporte automatizado ou semi-automatizado ao processo e aos métodos; quando as ferramenta se integram tem-se um sistema denominado CASE (Computer Aided Software Engineering)

# Engenharia de Software

## Uma Visão Genérica: 3 Fases

- Definição - “o que”
  - Engenharia do Sistema
  - Planejamento do Projeto
  - Análise de Requisitos
- Desenvolvimento - “como”
  - Projeto
  - Geração do Código
  - Teste
- Manutenção



## Atividades Guarda-Chuva

- Controle e Rastreamento do Projeto
- Revisões Técnicas Formais
- Garantia de Qualidade
- Gerenciamento de Configuração
- Produção e Preparação de Documentos
- Gerenciamento de Reusabilidade
- Medição
- Gerenciamento de Risco

# O Processo de Software

## Estrutura Comum de Processo

### Atividades da estrutura

#### Conjuntos de Tarefas

Tarefas

Marcos, Docs. Liberáveis

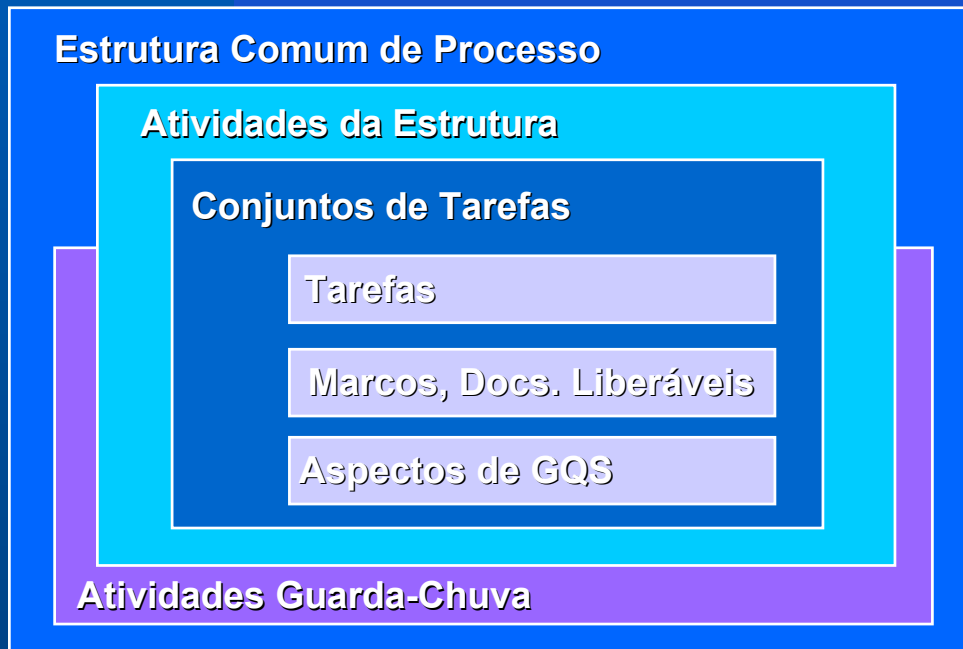
Aspectos de GQS

### Atividades Guarda-Chuva

## Estrutura Comum de Processo:

é estabelecido através da definição de um pequeno número de atividades que são aplicáveis a qualquer projeto de software, independentemente de seu tamanho ou complexidade

# O Processo de Software



## Conjuntos de Tarefas:

cada um é uma coleção de tarefas da Eng. Software - marcos de projeto, produtos de software e docs. liberáveis e pontos de garantia de qualidade - que permite que a estrutura de atividades seja adaptada às características do projeto e à equipe



# O Processo de Software

## Estrutura Comum de Processo

### Atividades da estrutura

#### Conjuntos de Tarefas

Tarefas

Marcos, Docs. Liberáveis

Aspectos de GQS

### Atividades Guarda-Chuva

## Atividades Guarda-Chuva:

garantia de qualidade de software, gerenciamento de configuração, medição - cobrem todo o processo e são independentes das atividades da estrutura

# O Processo de Software



Ênfase Atual



*“Maturidade do  
Processo”*




CMM (SEI)

# Modelos de Processo de Software

processos  
métodos  
ferramentas  
fases genéricas

devem incorporar  
uma *estratégia* de  
desenvolvimento



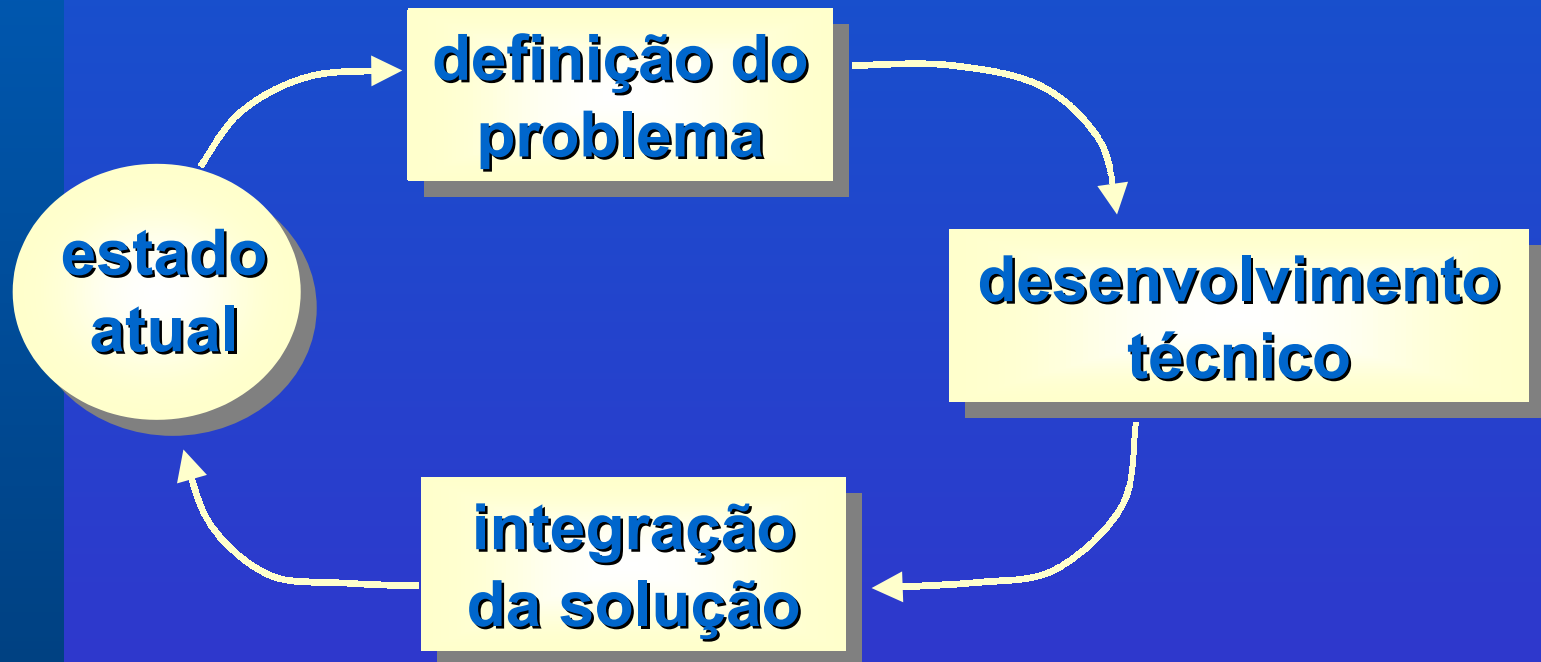
**Modelo de Processo**  
**Paradigma de Engenharia de Software**  
**Ciclo de Vida**

# Modelo de Processo de Software

- É escolhido com base:
  - na natureza do projeto e da aplicação
  - nos métodos e ferramentas a serem utilizados
  - nos controles e produtos que precisam ser entregues

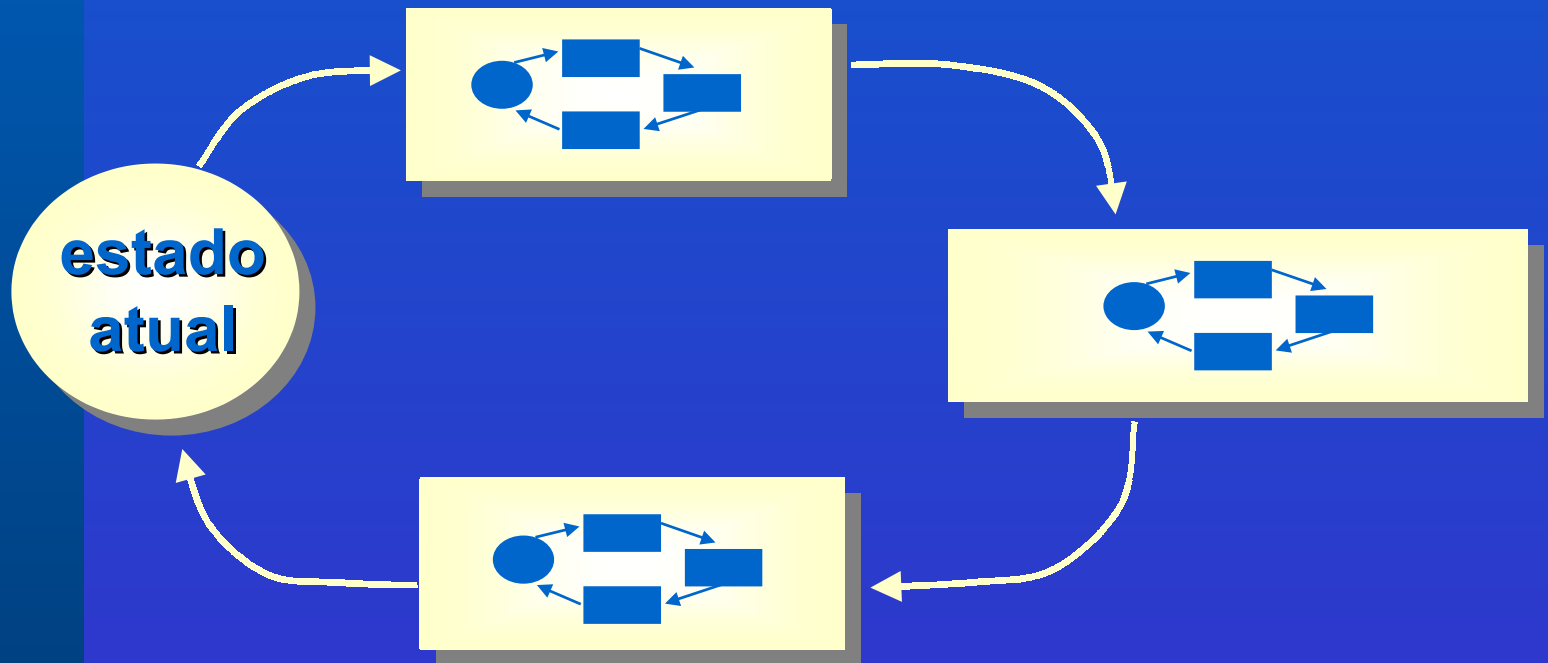
# Modelo de Processo de Software

Todo desenvolvimento de software pode ser caracterizado como um ciclo de solução do problema



# Modelo de Processo de Software

O ciclo de solução pode ser usado em diferentes níveis de resolução



# Modelos de Processo de Software

## Alguns Modelos de Processo:

- **Modelo Seqüencial Linear**
- **Modelo de Prototipação**
- **Modelo RAD**
- **Modelos Evolucionários**
  - **Incremental**
  - **Espiral**
  - **Montagem de Componente**
  - **Desenvolvimento Concorrente**
- **Modelo de Métodos Formais**
- **Técnicas de 4ª Geração**

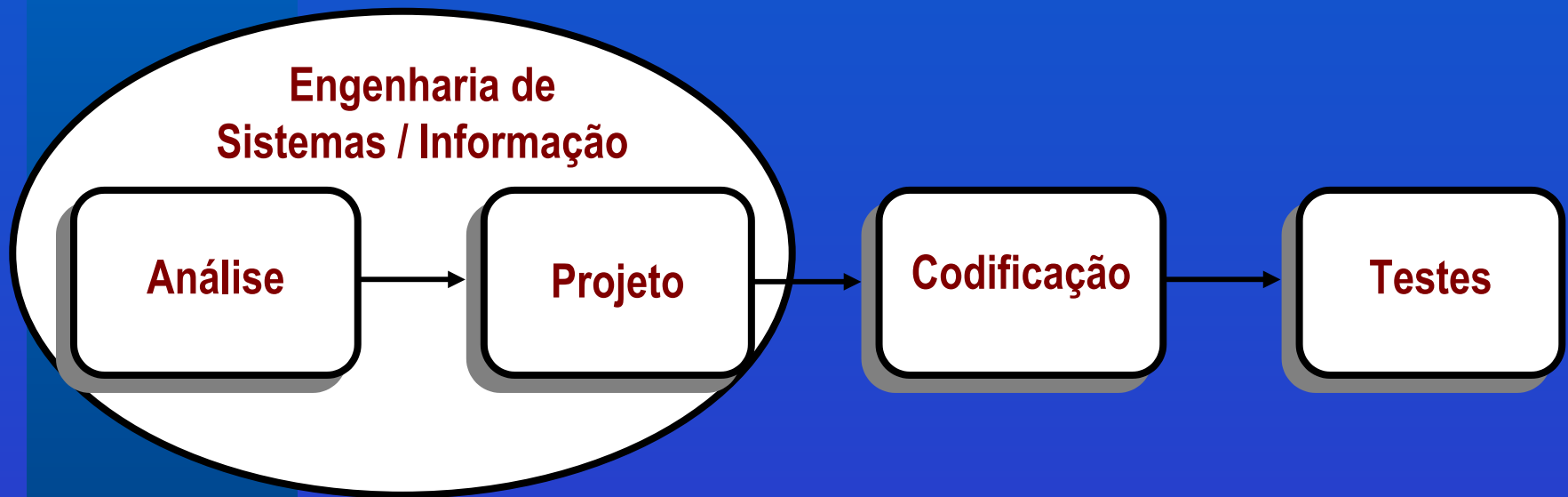
# Modelo Seqüencial Linear

## Ciclo de Vida Clássico ou Modelo Cascata

- modelo mais antigo e o mais amplamente usado da engenharia de software
- modelado em função do ciclo da engenharia convencional
- requer uma abordagem sistemática, seqüencial ao desenvolvimento de software

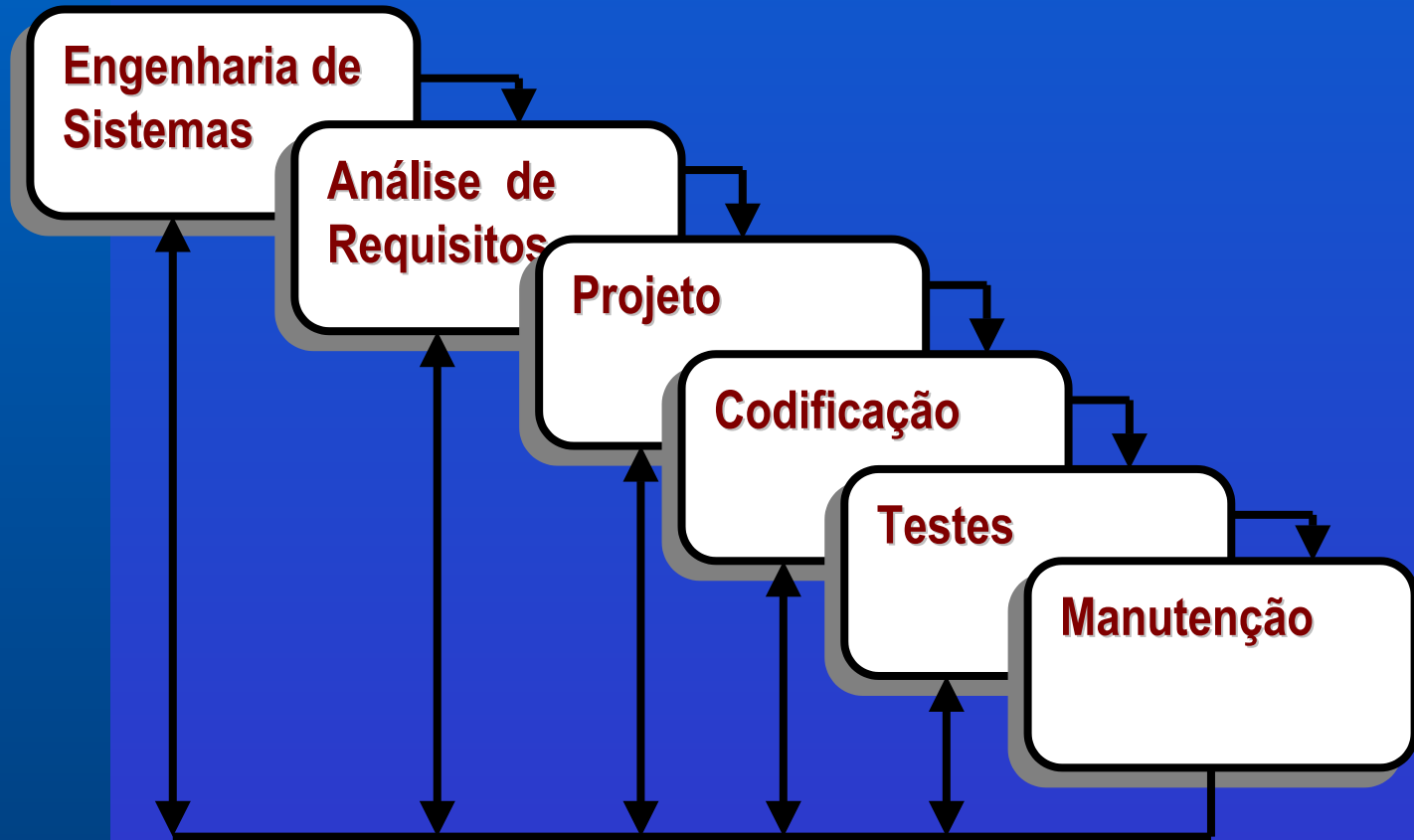


# Modelo Seqüencial Linear



muitas organizações que usam esse modelo,  
aplicam-no de forma estritamente linear

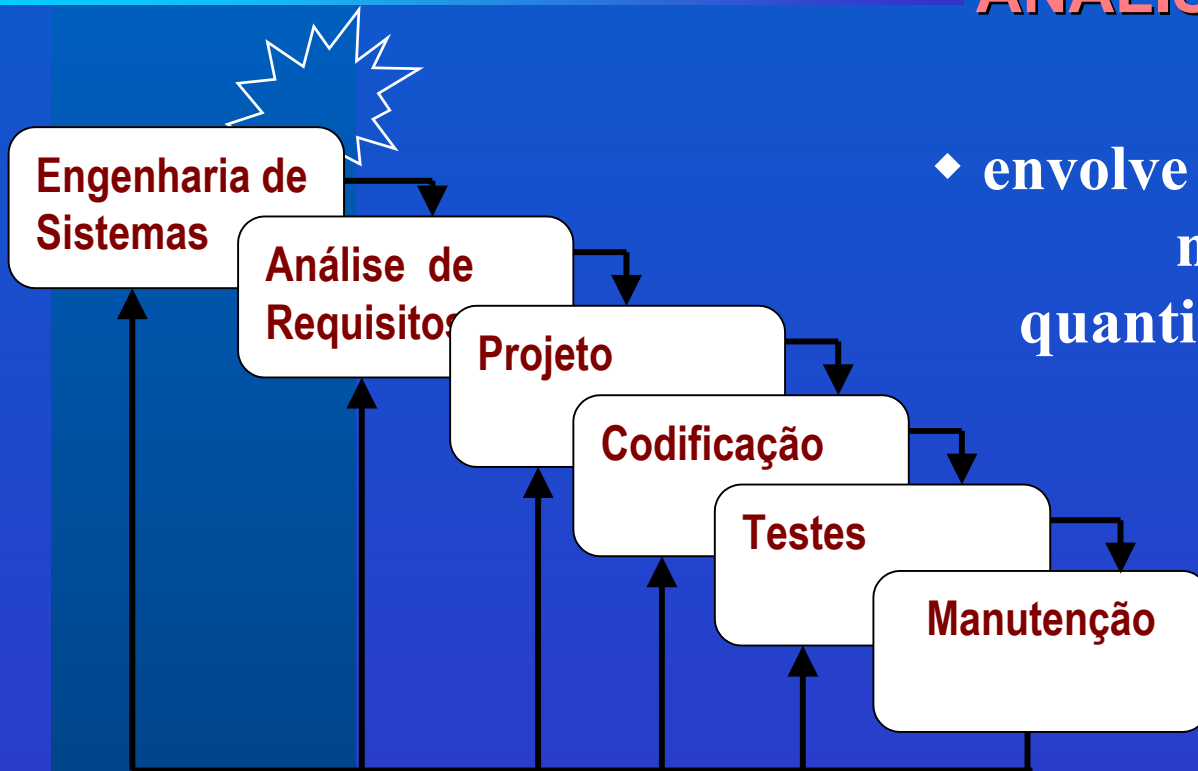
# Modelo Seqüencial Linear



modelo original, proposto por Royce, prevê feedback

# Atividades do Modelo Seqüencial Linear

## ANÁLISE E ENGENHARIA DE SISTEMAS

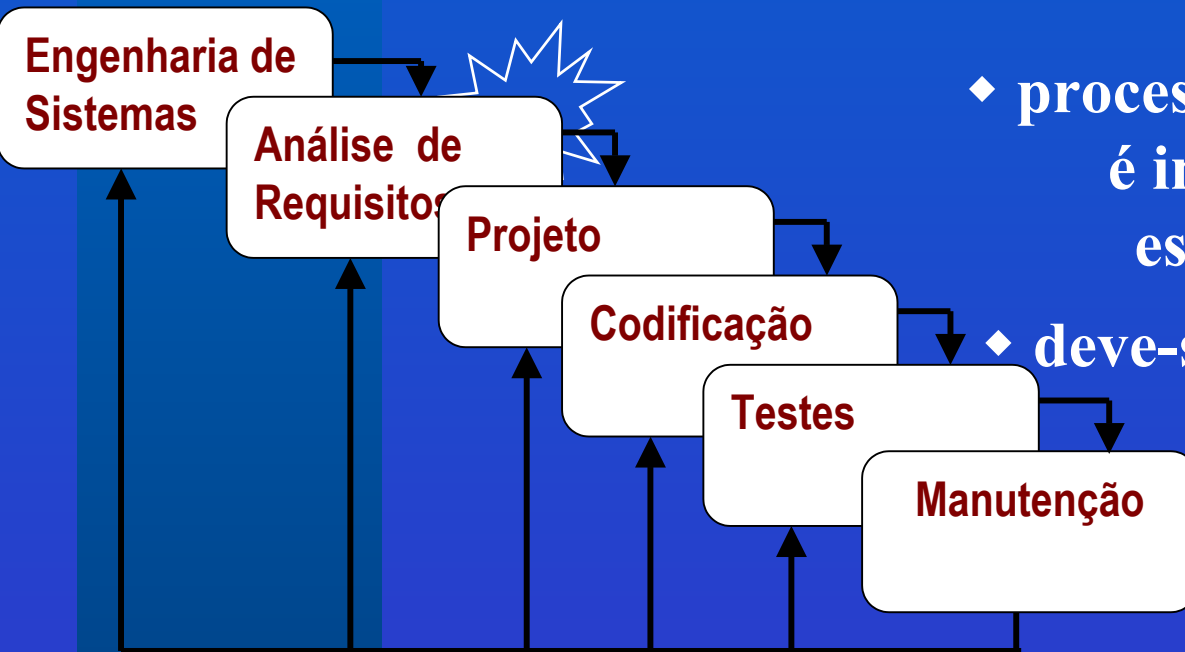


- ♦ envolve a coleta de requisitos em nível do sistema, pequena quantidade de projeto e análise de alto nível

- ♦ visão essencial quando o software deve fazer interface com outros elementos (hardware, pessoas e banco de dados)

# Atividades do Modelo Seqüencial Linear

## ANÁLISE DE REQUISITOS DE SOFTWARE



- ♦ processo de coleta dos requisitos é intensificado e concentrado especificamente no software
- ♦ deve-se compreender o domínio da informação, a função, desempenho e interfaces exigidos
- ♦ os requisitos (para o sistema e para o software) são documentados e revistos com o cliente

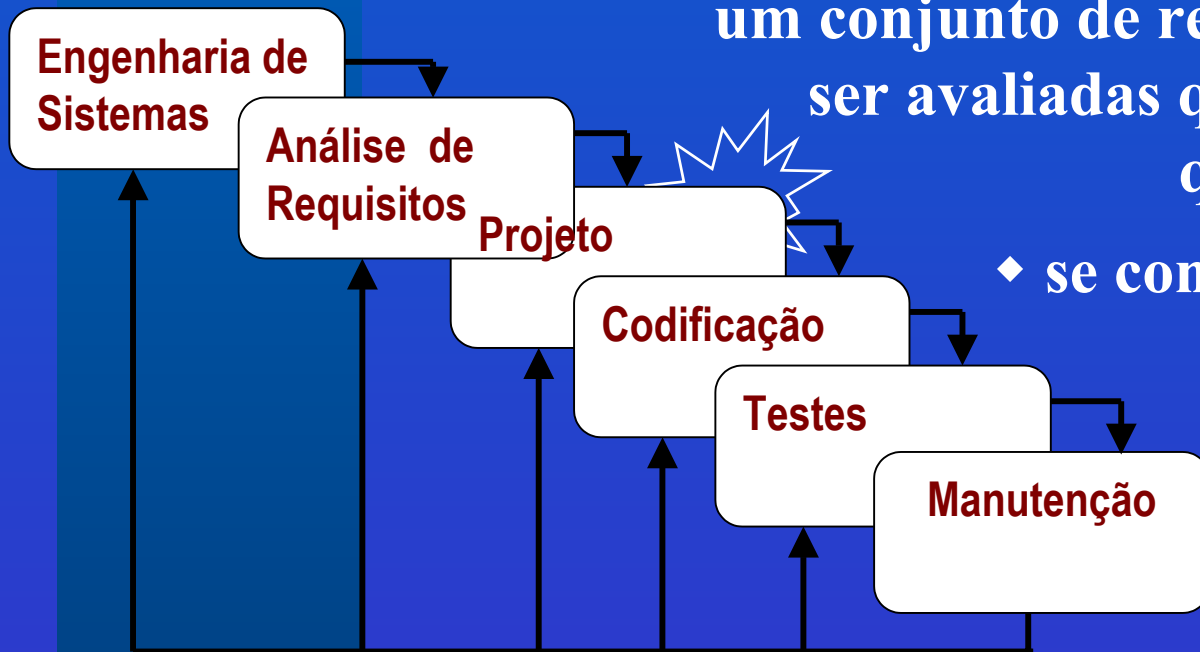
# Atividades do Modelo Seqüencial Linear

## PROJETO

♦ tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a codificação se inicie

♦ se concentra em 4 atributos do programa:

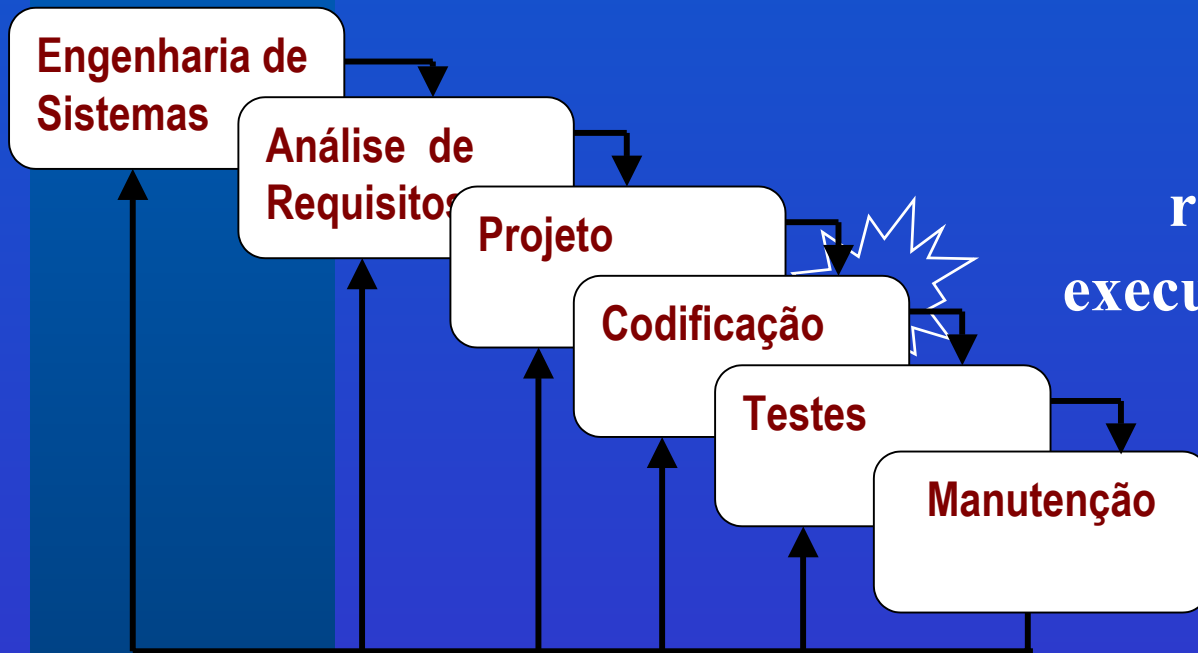
- *Estrutura de Dados,*
- *Arquitetura de Software,*
- *Detalhes Procedimentais e*
- *Caracterização de Interfaces*



# Atividades do Modelo Seqüencial Linear

## CODIFICAÇÃO

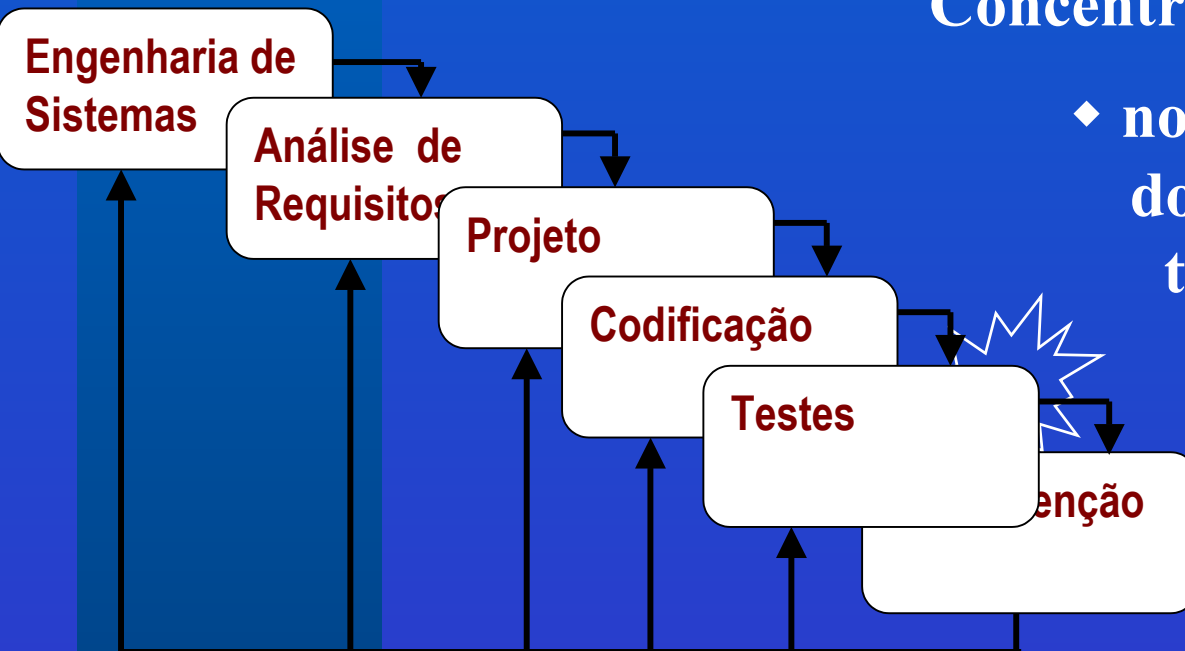
- ♦ tradução das representações do projeto para uma linguagem “artificial” resultando em instruções executáveis pelo computador



# Atividades do Modelo Seqüencial Linear

## TESTES

Concentram-se:

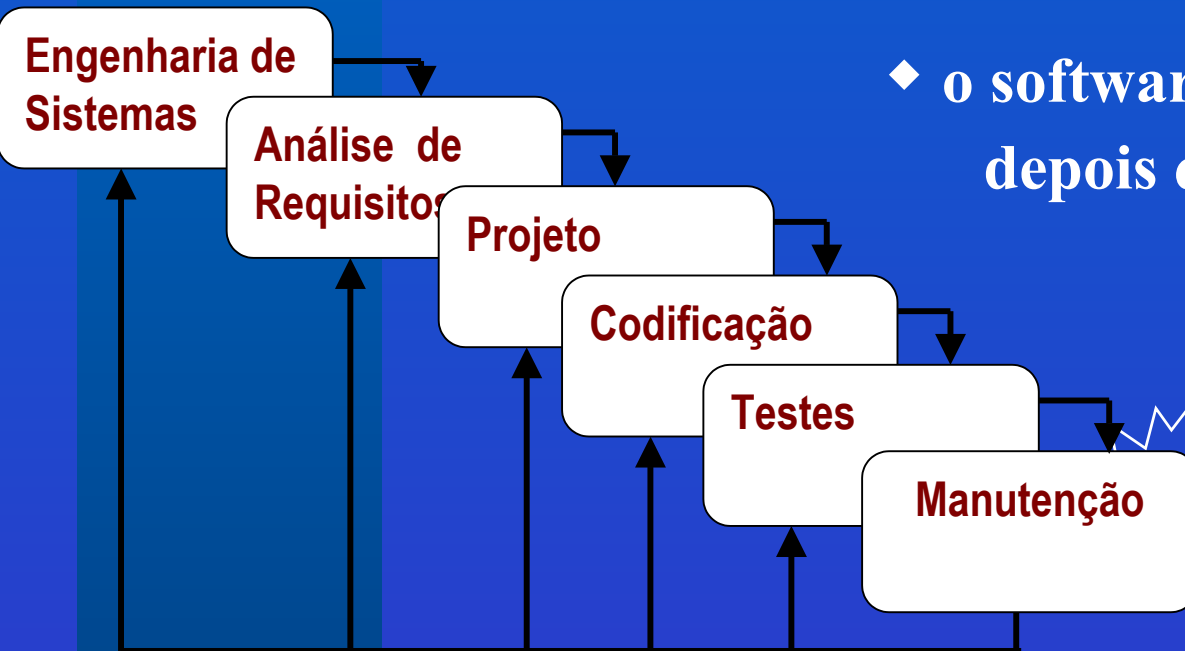


♦ nos **aspectos lógicos internos** do software, garantindo que todas as instruções tenham sido testadas

♦ nos **aspectos funcionais externos**, para descobrir erros e garantir que a entrada definida produza resultados que concordem com os esperados.

# Atividades do Modelo Seqüencial Linear

## MANUTENÇÃO



♦ o software deverá sofrer mudanças depois que for entregue ao cliente

♦ causas das mudanças:  
*erros, adaptação do software para acomodar mudanças em seu ambiente externo e exigência do cliente para acréscimos funcionais e de desempenho*



# Problemas com o Modelo Seqüencial Linear

- 💣 projetos reais raramente seguem o fluxo seqüencial que o modelo propõe
- 💣 logo no início é difícil estabelecer explicitamente todos os requisitos. No começo dos projetos sempre existe uma incerteza natural
- 💣 o cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento
- 💣 muitas vezes os desenvolvedores ficam ociosos desnecessariamente, devido a estados bloqueadores (quando existem tarefas dependentes, membros da equipe devem aguardar que outros terminem)

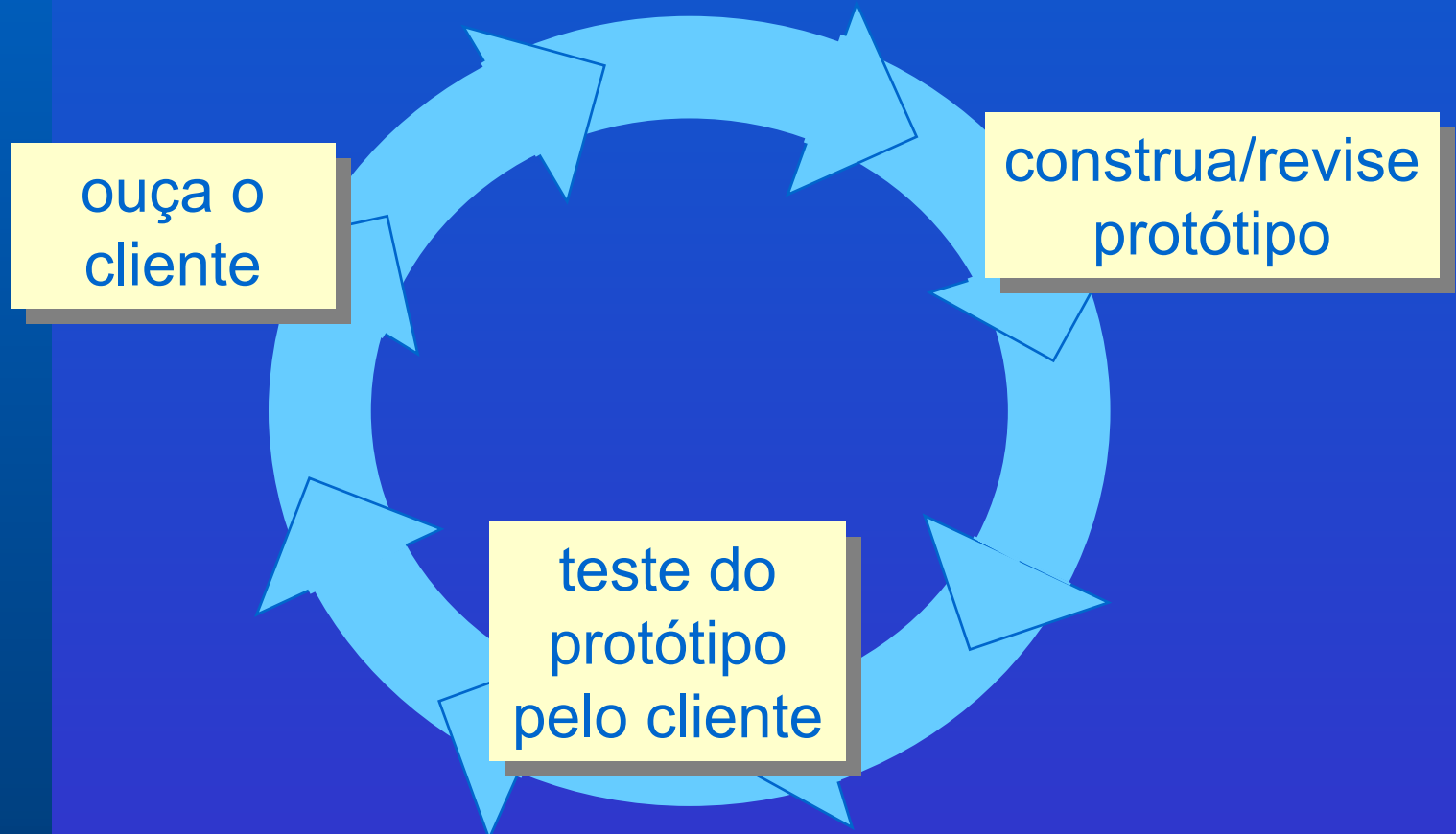
## Modelo Seqüencial Linear (comentários)

*Embora o Modelo Seqüencial Linear ou Ciclo de Vida Clássico tenha fragilidades, ele é significativamente melhor do que uma abordagem casual ao desenvolvimento de software*

# Prototipação

- ⇒ processo que possibilita que o desenvolvedor crie um modelo do software que deve ser construído.
- ⇒ idealmente, o modelo (**protótipo**) serve como um mecanismo para identificar os requisitos de software.
- ⇒ apropriado para quando o cliente definiu um conjunto de objetivos gerais para o software, mas não identificou requisitos de entrada, processamento e saída com detalhes.

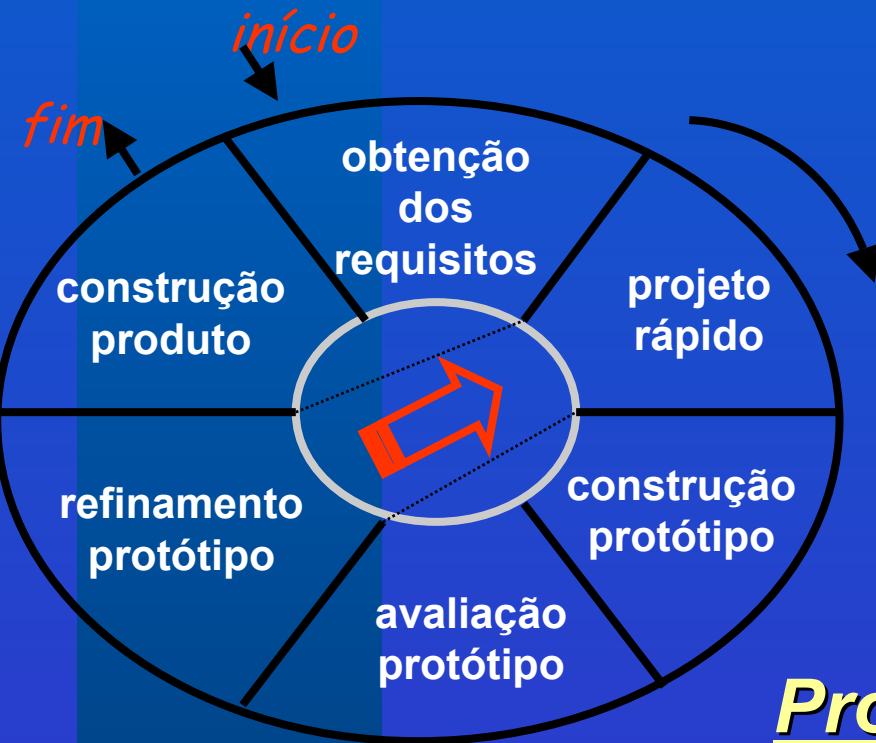
# Prototipação



# Prototipação



# Atividades da Prototipação



## **Obtenção dos Requisitos:**

desenvolvedor e cliente definem os objetivos gerais do software, identificam quais requisitos são conhecidos e as áreas que necessitam de definições adicionais

**Projeto Rápido:** representação dos aspectos do software que são visíveis ao usuário (abordagens de entrada e formatos de saída)

# Atividades da Prototipação



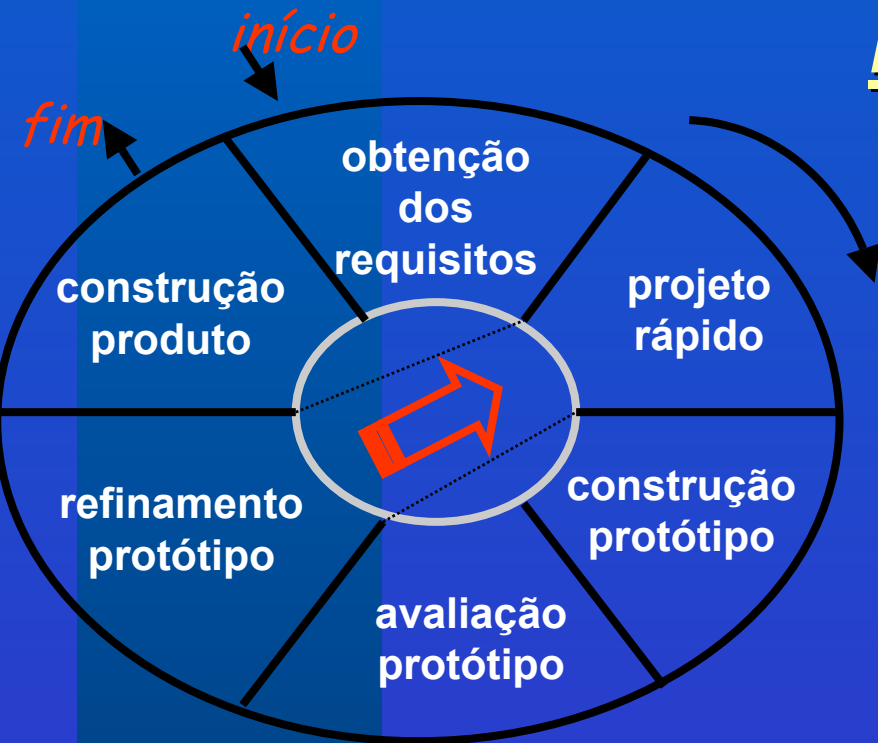
## **Construção Protótipo:**

implementação rápida do projeto

## **Avaliação do Protótipo:**

cliente e desenvolvedor avaliam o protótipo

# Atividades da Prototipação



## Refinamento dos Requisitos:

cliente e desenvolvedor refinam os requisitos do software a ser desenvolvido.

Ocorre neste ponto um *processo de iteração* que pode conduzir à 1a. atividade até que as necessidades do cliente sejam satisfeitas e o desenvolvedor compreenda o que precisa ser feito.



# Atividades da Prototipação



## **Construção Produto:**

identificados os requisitos, o protótipo deve ser descartado e a versão de produção deve ser construída considerando os critérios de qualidade.

# Problemas com a Prototipação

- 💣 cliente não sabe que o software que ele vê não considerou, durante o desenvolvimento, a qualidade global e a manutenibilidade a longo prazo. Não aceita bem a idéia de que a versão final do software vai ser construída e "força" a utilização do protótipo como produto final.

# Problemas com a Prototipação

- 💣 desenvolvedor freqüentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo. Depois de um tempo ele se familiariza com essas escolhas, e esquece que elas não são apropriadas para o produto final.

# Prototipação (comentários)

Ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente ✓

A chave é definirem-se as regras do jogo logo no começo ✓

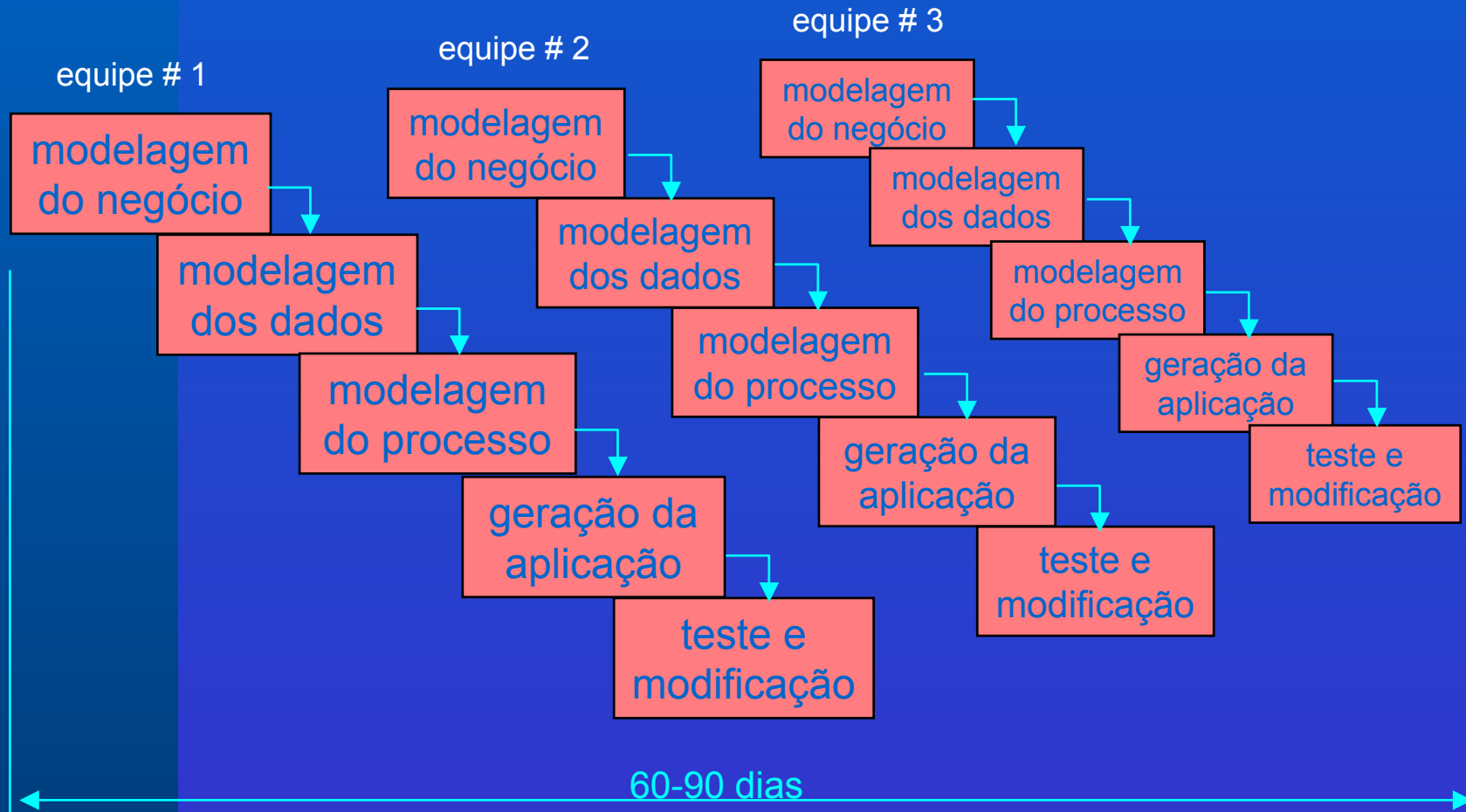
O cliente e o desenvolvedor devem ambos concordar que o protótipo seja construído para servir como um mecanismo a fim de definir os requisitos ✓

# Modelo RAD

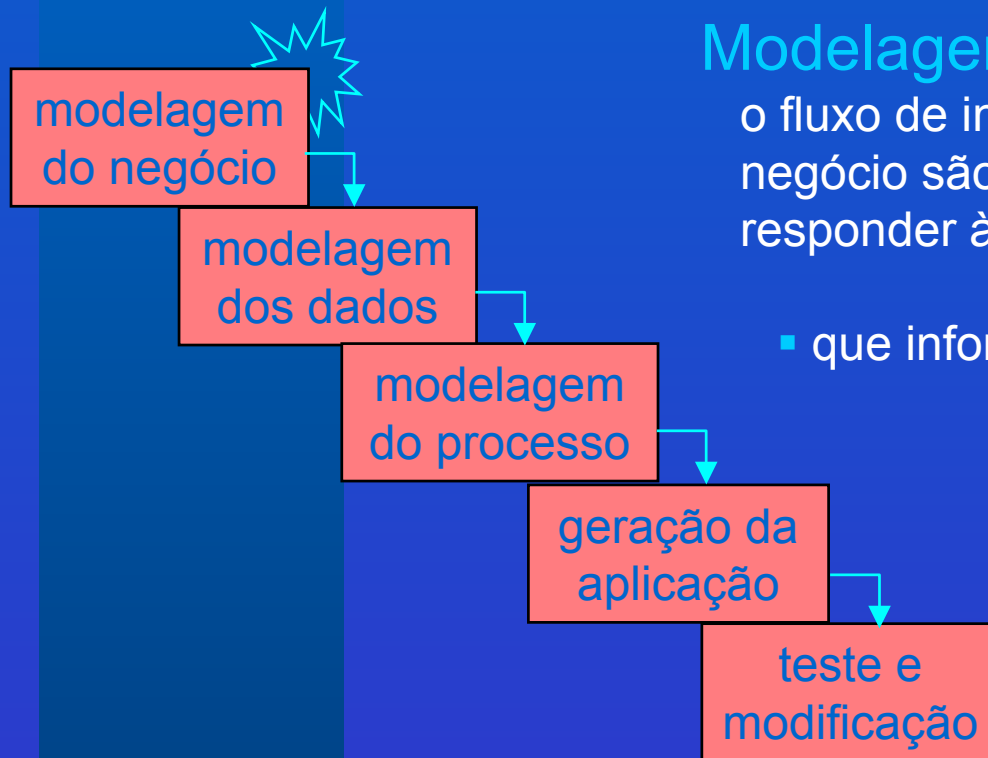
## (Rapid Application Development)

- ➡ É o modelo seqüencial linear mas que enfatiza um desenvolvimento extremamente rápido
- ➡ A “alta velocidade” é conseguida através de uma abordagem de construção baseada em componentes
- ➡ Usado quando os requisitos são bem definidos e o escopo do sistema é restrito
- ➡ Abordagem usada principalmente para aplicações de SI

# Modelo RAD



# Atividades do Modelo RAD

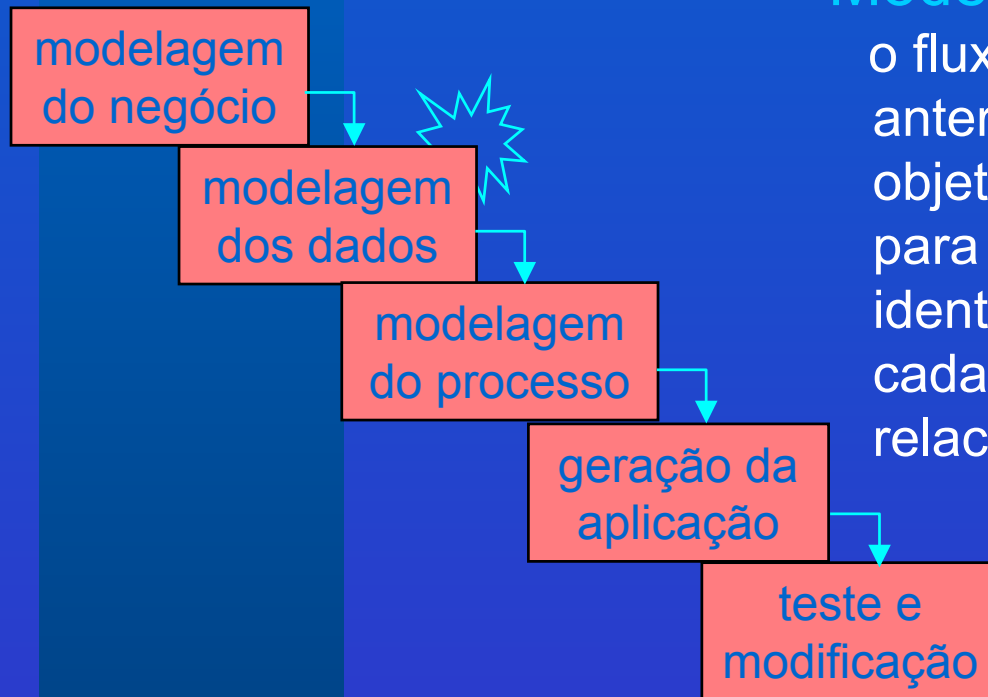


## Modelagem do negócio:

o fluxo de informação entre as funções do negócio são modeladas de maneira a responder às questões:

- que informação dirige o processo do negócio?
  - que informação é gerada?
  - quem gera a informação?
- para onde a informação vai?
  - quem a processa?

# Atividades do Modelo RAD

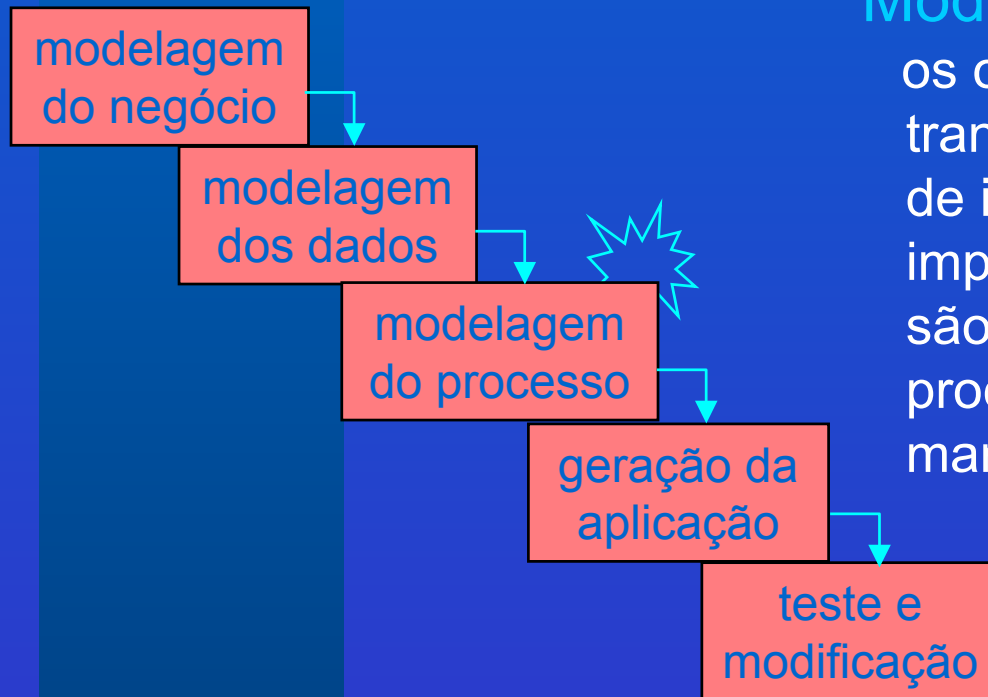


## Modelagem dos dados:

o fluxo de informação definido na fase anterior é refinado em um conjunto de objetos de dados que são necessários para dar suporte ao negócio; são identificadas as características de cada objeto e são definidos seus relacionamentos



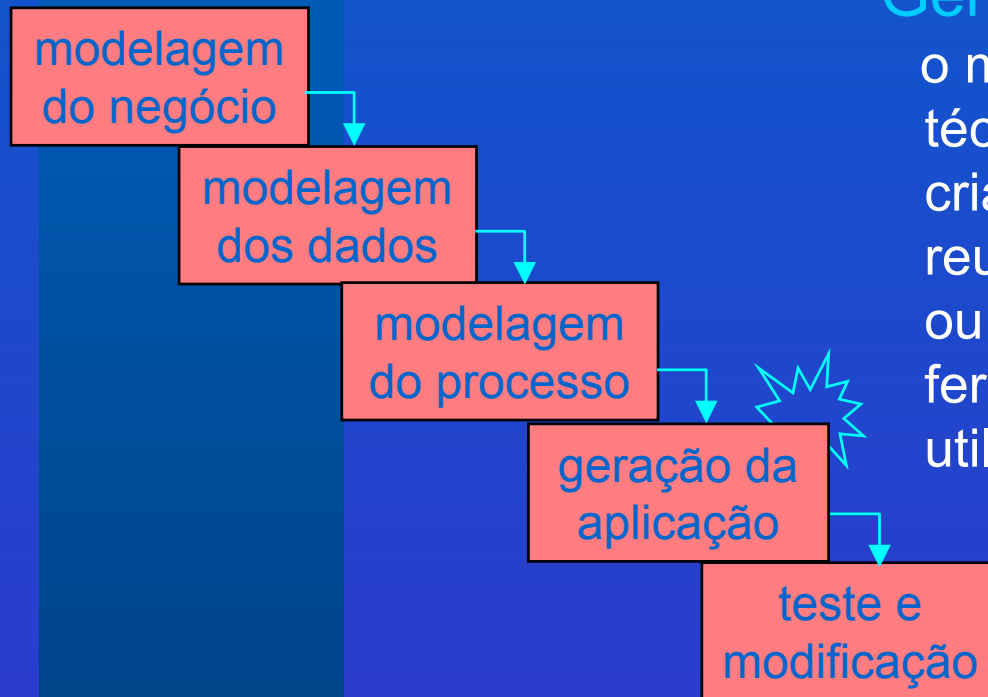
# Atividades do Modelo RAD



## Modelagem do processo:

os objetos de dados definidos são transformados para se obter o fluxo de informação necessário para implementar uma função do negócio; são criadas as descrições dos processamentos necessários para manipular esses objetos de dados

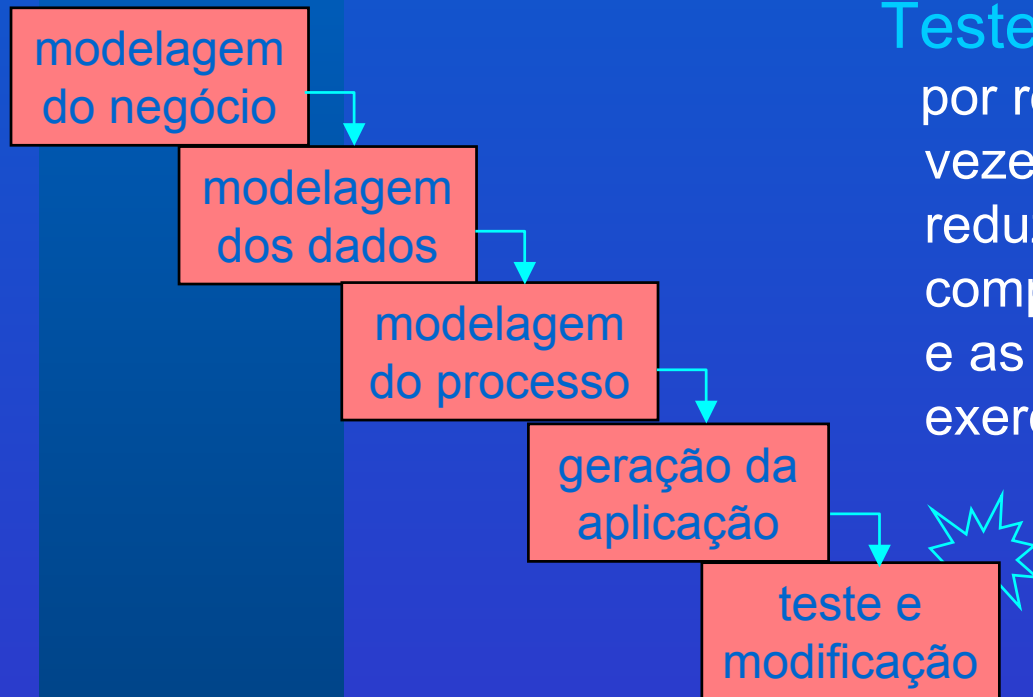
# Atividades do Modelo RAD



## Geração da aplicação:

o modelo RAD assume o uso de técnicas de 4a. geração; ao invés de criar software de forma convencional, reusa componentes quando possível ou cria componentes reutilizáveis; ferramentas automatizadas são utilizadas para gerar software

# Atividades do Modelo RAD



## Teste e modificação:

por reutilizar componentes, muitas vezes eles já foram testados, o que reduz o tempo de teste; os novos componentes devem ser testados e as interfaces devem ser exercitadas

# Modelo RAD

## Quando usar?

- > as restrições de tempo impostas pelo projeto demandam um escopo de escala
- > quando a aplicação pode ser modularizada de forma que cada grande função possa ser completada em menos de 3 meses
- > cada grande função pode ser alocada para uma equipe distinta e, depois são integradas para formar o todo


# Problemas com o Modelo RAD

- > para projetos escaláveis, mas grandes, o RAD requer recursos humanos suficientes para criar um número adequado de equipes
- > RAD requer um comprometimento entre desenvolvedores e clientes para que as atividades possam ser realizadas rapidamente e o sistema seja concluído em um tempo abreviado
- > se o comprometimento for abandonado por qualquer das partes, o projeto falhará
- > não é apropriado quando os riscos técnicos são grandes

# Modelos de Processo Evolucionários

- usado quando o deadline não é adequado para o desenvolvimento do software; a data de término não é realística
- uma versão limitada pode ser introduzida para atender à competitividade e pressões do negócio
- são liberados “produtos core”
- os detalhes e extensões ainda devem ser definidos
- Ex: editor de texto

# Modelos de Processo Evolucionários

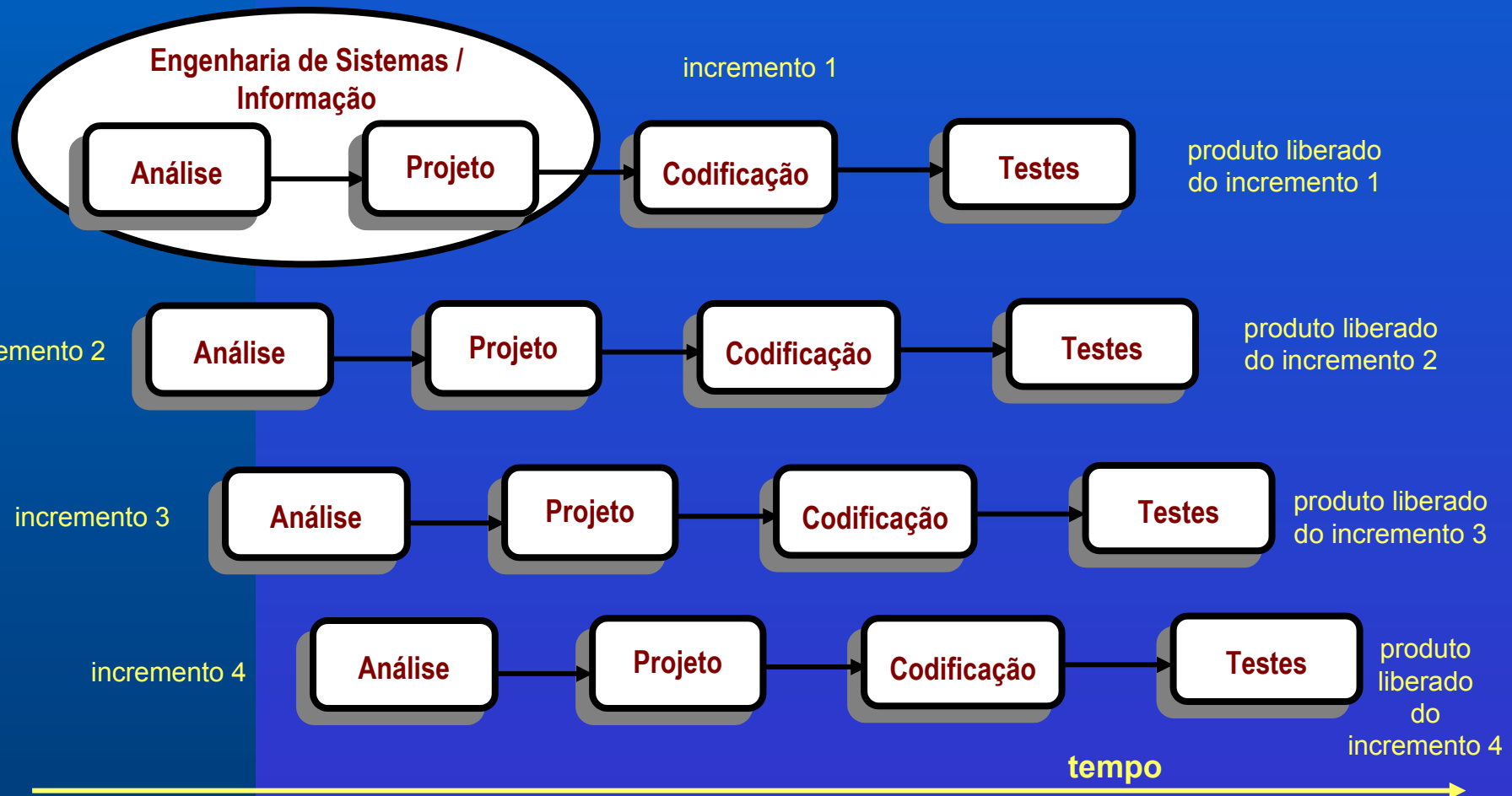
- 
- Incremental
  - Espiral
  - Montagem de Componentes
  - Desenvolvimento Concorrente

# Modelo Incremental

- combina elementos do Modelo Linear com a filosofia da Prototipação
- aplica seqüências lineares numa abordagem de “saltos” à medida que o tempo progride
- Cada seqüência linear produz um incremento do software (proc. de texto)
- O processo se repete até que um produto completo seja produzido
- Difere da Prototipação pois a cada incremento produz uma versão operacional do software



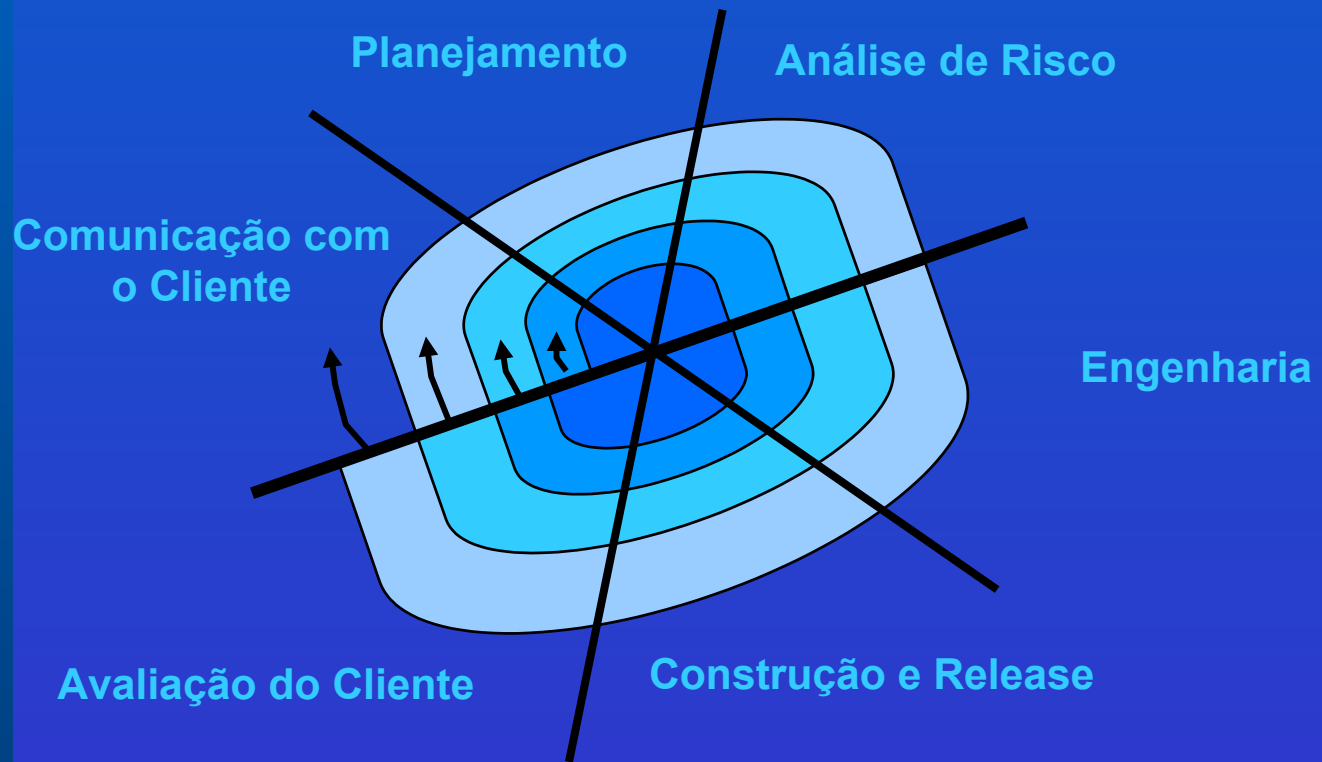
# Modelo Incremental



# Modelo Espiral

- ➡ engloba a natureza iterativa da **Prototipação** com os aspectos sistemáticos e controlados do **Modelo Linear**
- ➡ fornece o potencial para o desenvolvimento rápido de versões incrementais do software
- ➡ nas primeiras iterações a versão incremental pode ser um modelo em papel ou um protótipo
- ➡ nas iterações mais adiantadas são produzidas versões incrementais mais completas e melhoradas

# Modelo Espiral



# Atividades do Modelo Espiral

## Comunicação com o cliente:

tarefas requeridas para estabelecer uma efetiva comunicação entre desenvolvedor e cliente

## Planejamento:

tarefas requeridas para definir recursos, referenciais de tempo e outras informações de projeto

## Análise de Risco:

tarefas requeridas para fazer levantamento de riscos técnicos e de gerenciamento

## Engenharia:

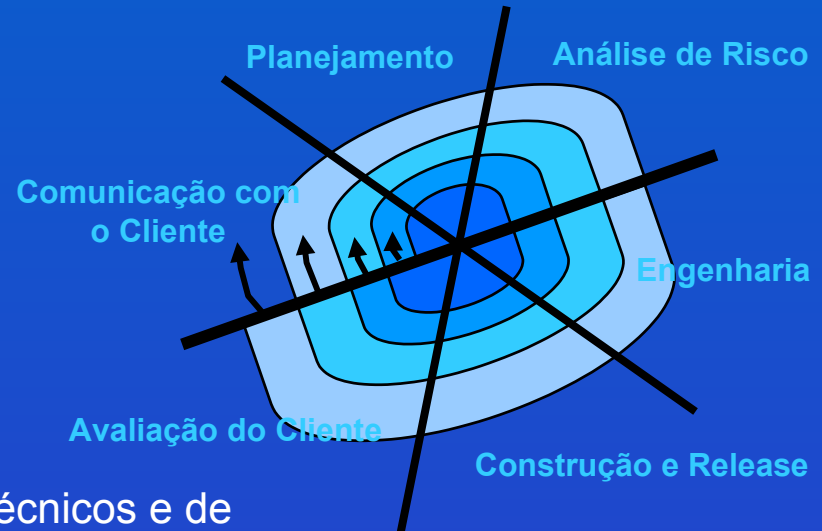
tarefas requeridas para construir uma ou mais representações da aplicação

## Construção e Release:

tarefas requeridas para construir, testar, instalar e dar suporte ao usuário (p.ex., documentação e treinamento)

## Avaliação do cliente:

tarefas requeridas para obter um feedback do cliente baseado na avaliação da representação do software criado durante a fase de engenharia e implementado durante a fase de instalação



# Modelo Espiral (comentários)

- ✎ é, atualmente, a abordagem mais realística para o desenvolvimento de software em grande escala
- ✎ usa uma abordagem que capacita o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva
- ✎ pode ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável
- ✎ exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso

# Modelo Espiral (comentários)

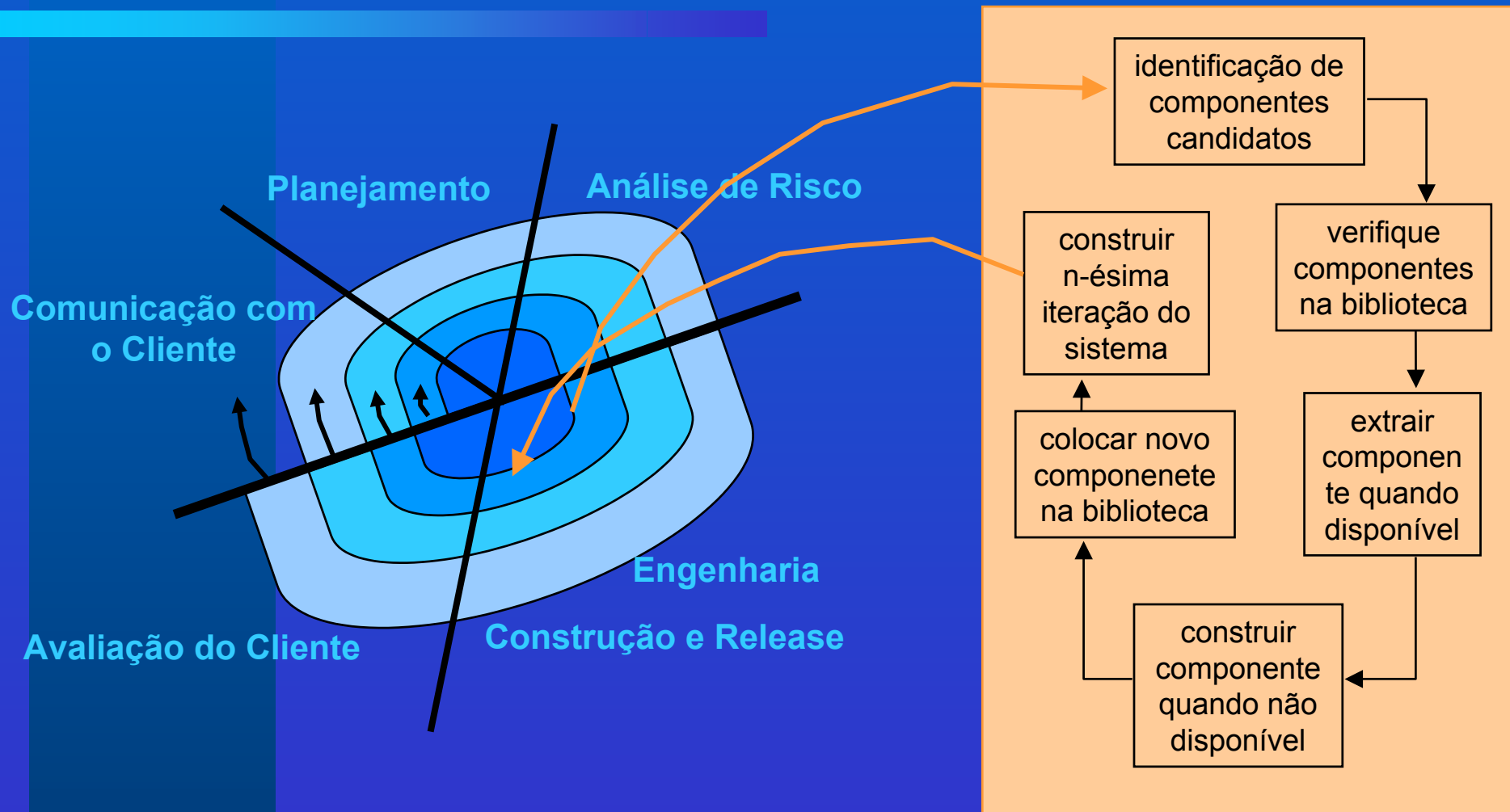
👉 o modelo é relativamente novo e não tem sido amplamente usado

Serão necessários alguns anos até que a eficácia desse modelo possa ser determinada com certeza absoluta.

# Modelo de Montagem de Componentes

- ➔ incorpora características de tecnologias Orientadas a Objetos no Modelo Espiral
- ➔ a atividade de Engenharia começa com a identificação de classes candidatas
- ➔ se a classe existe, ela será reutilizada
- ➔ se a classe não existe, ela será desenvolvida nos moldes do paradigma de Orientação a Objetos

# Modelo de Montagem de Componentes





# Modelo de Desenvolvimento Concorrente

- ➡ é representado como uma série de grandes atividades técnicas, tarefas e seus estados associados
- ➡ ele define uma série de eventos que podem disparar transições de um estado para outro, para cada uma das atividades da engenharia de software
- ➡ é frequentemente usado como um paradigma para o desenvolvimento de aplicações Cliente/Servidor
- ➡ pode ser aplicado a todo tipo de desenvolvimento de software e fornece uma visão exata de como está o estado do projeto

# Modelo de Desenvolvimento Concorrente

Atividade de Análise



# Modelo de Métodos Formais

- Compreende um conjunto de atividades que determinam uma especificação matemática para o software
- Uma variante dessa abordagem é denominada *Engenharia de Software Cleanroom*
- Utilizando métodos formais eliminam-se muitos problemas encontrados nos outros modelos, como p.ex., ambigüidade, incompletitude e inconsistência, que podem ser corrigidas mais facilmente de forma não *ad hoc* mas através de análise matemática
- Promete o desenvolvimento de software livre de defeitos

# Modelo de Métodos Formais (comentários)

- Atualmente esse modelo consome muito tempo e é muito caro
- Como poucos desenvolvedores possuem o background necessário para utilizá-lo, são requeridos muitos cursos e treinamentos
- É difícil usar tais modelos como meio de comunicação com a maioria dos clientes

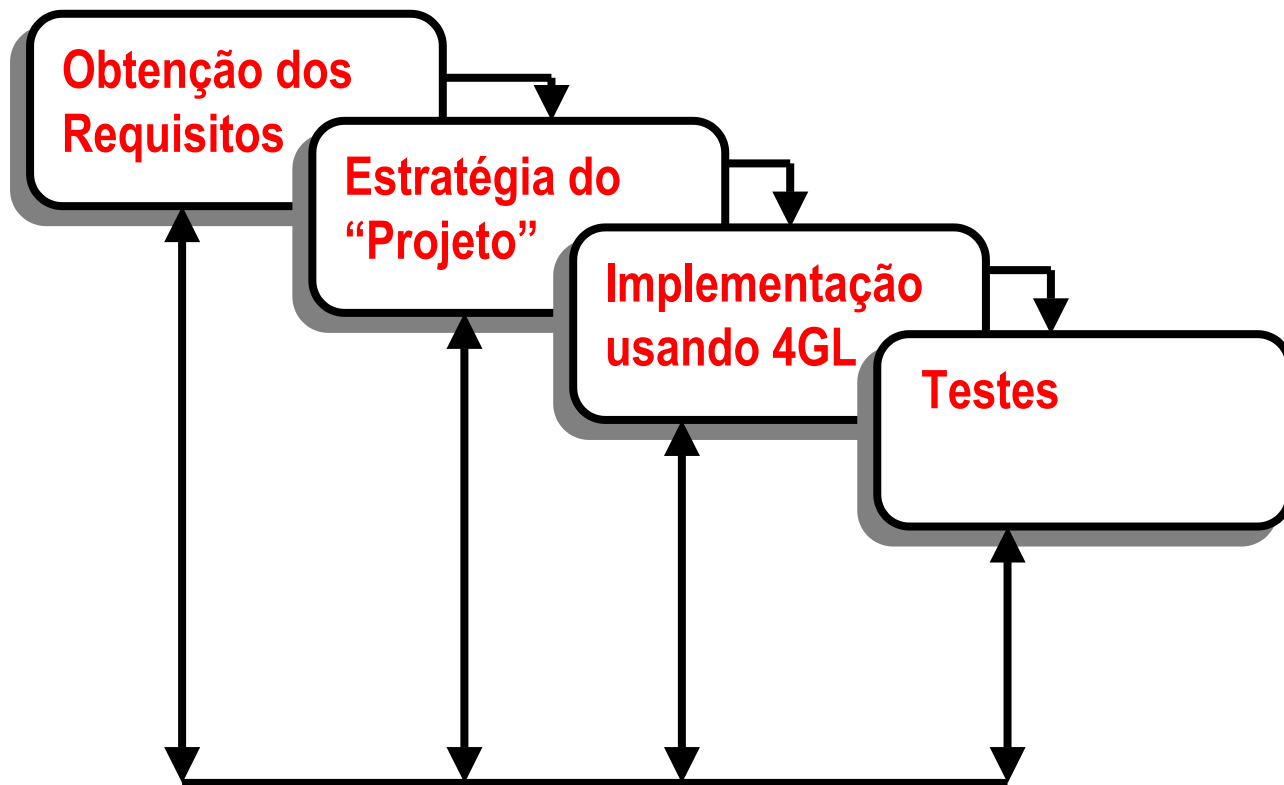
# Técnicas de 4ª Geração

Concentra-se na capacidade de se especificar o software a uma máquina em um nível que esteja próximo à linguagem natural.

Engloba um conjunto de ferramentas de software que possibilitam que:

- ⇒ o sistema seja especificado em uma linguagem de alto nível e
- ⇒ o código fonte seja gerado automaticamente a partir dessas especificações

# Técnicas de 4ª Geração



# Ferramentas do ambiente de desenvolvimento de software de 4GL

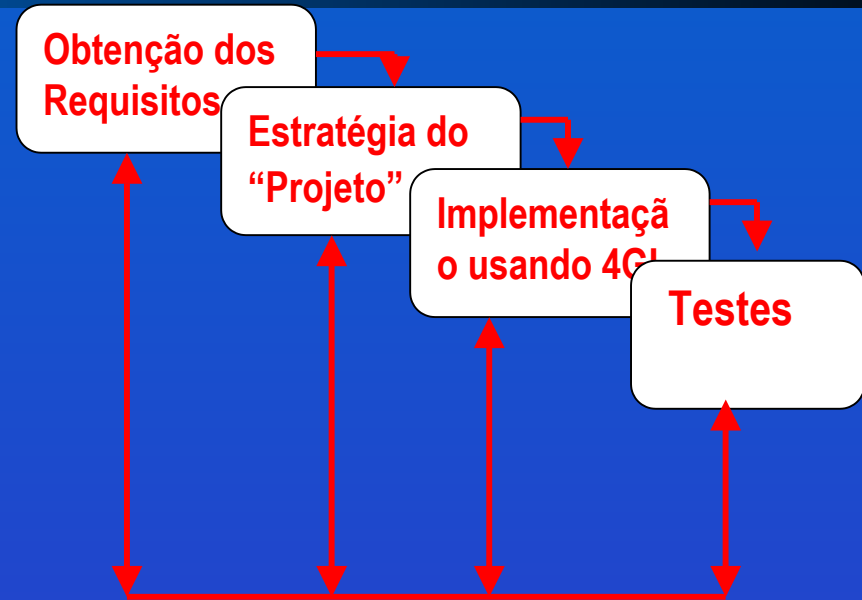
O ambiente de desenvolvimento de software que sustenta o ciclo de vida de 4ª geração inclui as ferramentas:

- ✓ *linguagens não procedimentais para consulta de banco de dados*
- ✓ *geração de relatórios*
- ✓ *manipulação de dados*
- ✓ *interação e definição de telas*
- ✓ *geração de códigos*
- ✓ *capacidade gráfica de alto nível*
- ✓ *capacidade de planilhas eletrônicas*

# Atividades das Técnicas de 4ª Geração

## 1. obtenção dos Requisitos:

o cliente descreve os requisitos os quais são traduzidos para um protótipo operacional



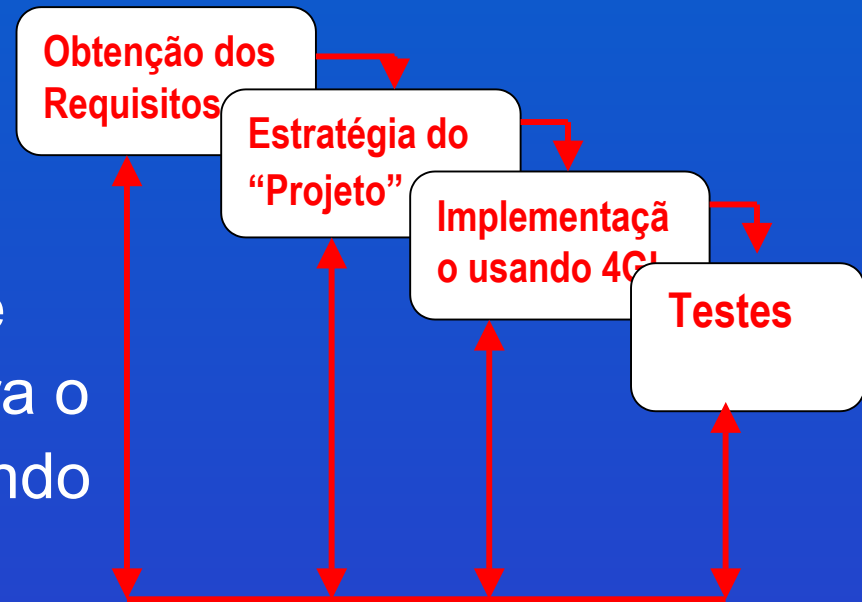
- ✘ o cliente pode estar inseguro quanto aos requisitos
- ✘ o cliente pode ser incapaz de especificar as informações de um modo que uma ferramenta 4GL possa consumir
- ✘ as 4GLs atuais não são sofisticadas suficientemente para acomodar a verdadeira "linguagem natural"



# Atividades das Técnicas de 4ª Geração

## 2. estratégia de "Projeto":

para pequenas aplicações é possível mover-se do passo de **Obtenção dos Requisitos** para o passo de **Implementação** usando uma **Linguagem de 4G**

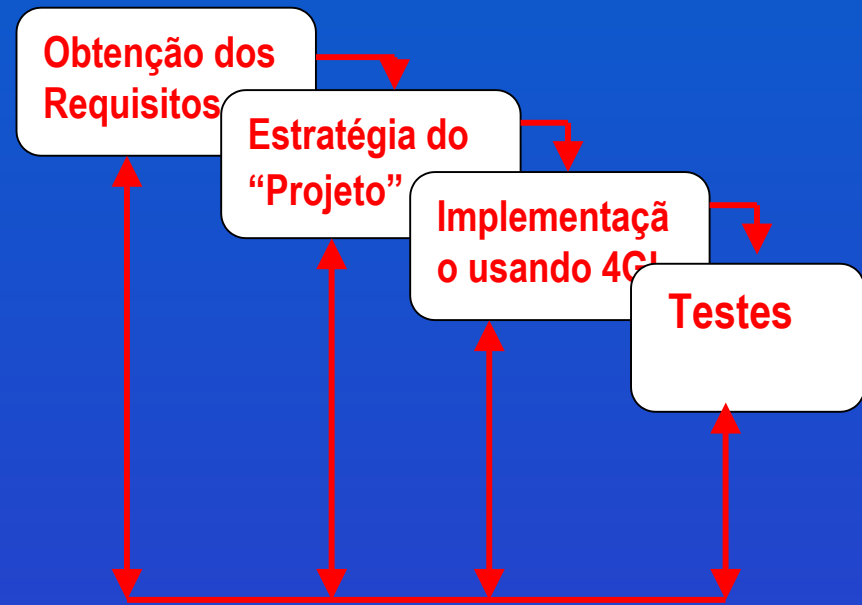


✖ para grandes projetos é necessário desenvolver uma estratégia de projeto. De outro modo ocorrerão os mesmos problemas encontrados quando se usa abordagem convencional (baixa qualidade)

# Atividades das Técnicas de 4ª Geração

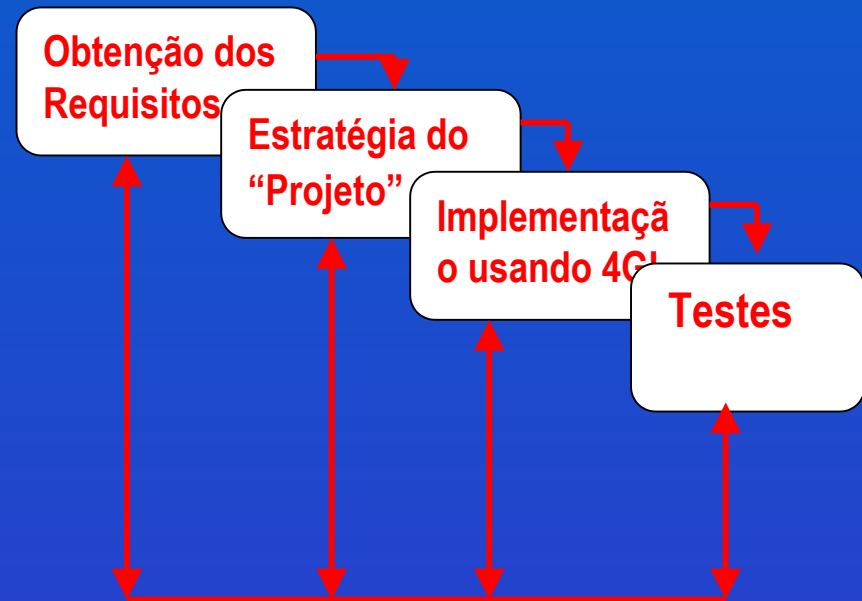
## 3. implementação usando

**4GL:** os resultados desejados são representados de modo que haja geração automática de código . Deve existir uma estrutura de dados com informações relevantes e que seja acessível pela 4GL



# Atividades das Técnicas de 4ª Geração

**4. teste:** o desenvolvedor deve efetuar testes e desenvolver uma documentação significativa. O software desenvolvido deve ser construído de maneira que a manutenção possa ser efetuada prontamente.



# Técnicas de 4ª Geração (comentários)

PROPONENTES: redução dramática no tempo de desenvolvimento do software (aumento de produtividade)

OPONENTES: as 4GL atuais não são mais fáceis de usar do que as linguagens de programação

💣 *o código fonte produzido é ineficiente*

💣 *a manutenibilidade de sistemas usando técnicas 4G ainda é questionável*

# Pontos Chaves

- Cada um dos modelos de processo possui potencialidades e fragilidades, porém possui uma série de fases genéricas comuns.
- Se o processo de software for fraco, o produto irá sofrer, mas uma ênfase somente no processo também é perigosa.



Deve-se tratar produto e processo como uma dualidade (algo que se complementam).